AD A050636

AD NO.
DDC FILE COPY

RADC-TR-76-387
Final Technical Report
December 1976

SOFTWARE DATA REPOSITORY STUDY

IIT Research Institute

Approved for public release;
distribution unlimited.

[illegible]
[illegible]

APPROVED: [illegible]

JOHN PALAZZO
Project Engineer

APPROVED: [illegible]

ROBERT D. KRUTZ, Col, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: [illegible]

[illegible]
[illegible]

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER 18 RADC-TR-76-387	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 SOFTWARE DATA REPOSITORY STUDY.		5. TYPE OF REPORT & PERIOD COVERED 9 Final Technical Rep[redacted]t. Jun[redacted]975 - Aug[redacted]76
		6. PERFORMING ORG. REPORT NUMBER 14 IITRI-E633Ø-28
7. AUTHOR(s) 10 Lorraine M./Duvall		8. CONTRACT OR GRANT NUMBER(s) 15 F3Ø6Ø2-75-C-Ø257 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS IIT Research Institute 10 West 35th Street Chicago IL 60616		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 63728F 16 5550816 17 Ø8
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISIS) Griffiss AFB NY 13441		12. REPORT DATE 11 Dec[redacted]976
		13. NUMBER OF PAGES 153 12 264 p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer:
John Palaimo (ISIS)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Repository	Software Development
Data Center	Information Systems
Software Engineering	Database Systems
Software Quality	Software Experience Data

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The purpose of the Software Data Repository is to upgrade the software development process through collection, analysis, and dissemination of software development experience.

A functional definition of the repository including a discussion of the inputs, processes, and outputs is presented in this report. The input processing and the requirements of an information system for storing and processing the data is discussed, along with a presentation of the recommendations for the

→ next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

175 35Ø

[illegible]

repository including the development and operation of a pilot facility and the expansion of this facility into a fully operational center.

PREFACE

This technical report presents the results of a study to develop the design for a Software Data Repository covering the period June 1975 to August 1976. This study was sponsored by the Information Sciences Division of the Rome Air Development Center (RADC), United States Air Force, under Contract No. F30602-75-C-0257. J. Palaimo of RADC was the Project Engineer.

In addition to the author of this report, project contributions were made by J. Church, K. Kogler, P. Llewellen, C. Proctor, R. Rebich, T. Ripley, P. Slawniak, P. Taska, and R. Wise from IIT Research Institute; and P. Belford and W. Whisenant from Computer Sciences Corporation.

Approved by	Submitted by
[illegible]	Lorraine Duvall
Ronald T. Anderson	Lorraine Duvall
Mgr., Reliability Section	Project Engineer
Electronics Research Division	IIT Research Institute

SOFTWARE DATA REPOSITORY

Table of Contents

Table of Contents (cont'd)

Table of Contents (cont'd)

LIST OF FIGURES

LIST OF TABLES

EVALUATION

This effort is part of a program, the goal of which is the implementation of a central repository for data and technical information for computer software technology and engineering. This center will provide tools and facilities to store, analyze, and disseminate authoritative scientific and technical information concerning all aspects of computer software and the software development process.

This effort fits into the RADC Technology Plan, particularly RADC TPO 11, Software Sciences Technology. The objective was to investigate areas relating to the design, implementation and operation of a central repository for software data, and to provide a prelininary design including functional description and recommendations for a pilot capability. All major objectives of this program were successfully addressed.

These results in concert with recommendations from a parallel study of software data collection procedures and problems are being incorporated into RADC plans for the implementation of a Data and Analysis Center for Software. The Software Data Collection Study was conducted by System Development Corp and is reported on in RADC-TR-76-329.

[illegible]

JOHN PALAIMO
Project Engineer

1. Introduction

1.1 Report Introduction

This report presents the results of a study to develop the design for a software data repository to collect, analyze, and disseminate software development experience information.

1.2 Background

Software development research has been hampered by the lack of data. Without the availability of historical data on the production of computer software, we lack the ability to predict costs of future development, to determine the best design, development, and testing methodologies, and the ability to effectively measure and predict the reliability of computer software systems.

In recognition of this need, RADC contracted for this study to establish the specifications for a software data repository for data and information exchange. This repository will serve the government/university/industrial community as the focal point for software development experience information. Data acquired from various sources will be stored in a centralized data base and made available to qualified users, analyses will be performed by computer software engineers and statisticians, and results will be produced in the form of publications to be disseminated to repository users.

The purposes of the software data repository are to upgrade the software development process through the collection, analysis and dissemination of software development experience information; to provide a better understanding of the nature, causes and effects of software failures; and to determine those factors that contribute to the productivity and cost of computer software development and maintenance.

The concept of a data center for the collection, analysis and dissemination of information in a specialized area is not new. As early as 1963, in "The Weinberg Report"[115] 400 Information Analysis Centers were identified. This panel recognized the need for centers that would distill technical information and would serve more than disseminators of documents. The 1970 version of the Directory of Federally - sponsored Centers lists 110 centers (COSATI Panel 6). As an example of these centers, there are presently eight DOD Information Analysis Centers which are administratively managed and funded by the Defense Supply Agency (DSA). They are responsible for the collection, evaluation, analysis and dissemination of scientific and technical information to the scientists, engineers and technicians they support. Among the specialized areas they deal with are hardware reliability, metals and ceramics, mechanical properties, thermophysical and electronic properties and machinability.

The generation of the design for the software data repository defined in this report was guided by the IITRI experience in developing and operating one of the above mentioned DOD Information Analysis Centers - The Reliability Analysis Center (RAC).[101, 106-114]

1.3 Report Organization

This report contains eight sections and three appendices.

Section 1 contains background information while Section 2 provides a functional definition of the Software Data Repository including a discussion of the inputs, processes, and outputs.

The requirements for processing the input data and documents is presented in Section 3 and Section 4 contains the requirements of an information system to manage the data including a discussion of the database characteristics and structure.

Section 5 contains discussions on data collection systems that automatically and manually collect software data, data entry hardware devices that are available on the market for consideration as tools to transform hard copy data to computer-readable form, alternative information system and database management system approaches, and alternative approaches for automating the document library.

The design for the Software Data Repository, including the development and operation of a pilot facility and the expansion to a fully operational center, is presented in Section 6. Section 7 contains a summary of the report recommendations and a bibliography is presented in Section 8.

Appendix A contains a glossary of database management terms. Presented in Appendix B are the results of a survey of the software development community to determine their interest in the repository. Appendix C contains definitions of software engineering terms as they appear in the literature.

2. Definition of the Software Data Repository

2.1 Introduction

This section presents and describes the functional model for the Software Data Repository. In pursuit of this model definition, the following subtasks were performed:

- Review of the software engineering literature
- Survey of the software development community
- Visits to software developers and attendance at software engineering conferences
- Meetings with RADC engineers
- Review of the literature on data center operations
- Observing the operation of the Reliability Analysis Center

2.2 Functional Model

The Hierarchy Input Process Output (HIPO) chart in Figure 2.1 provides a graphical representation of the functional model of the Software Data Repository (SDR). The principal inputs to the SDR include production/development reports that contain software experience data and documents that contain textual information on the software development process. The data and documents form the basis for a document library and the two computer databases that serve as inputs for subsequent updating and processing.

The major SDR processes include the acquisition of data and documents, the evaluation and summarization of these inputs, the production of reports and data subsets, the performance of engineering analysis, data analysis, and software model development.

The major outputs are the updated databases and document library, data subsets for modeling efforts, published reports containing results of analysis, and answers to inquiries and consulting actions.

As the document library and the two major databases form the heart of the repository, the information content of each is first discussed.

2.3 Information Content

There are two major classes of information that form the inputs to the SDR. The first class consists of production/development information relating to the developmental characteristics of producing and monitoring computer software. The second class consists of textual or expository information relating to software development. From these documents the Document Description Database, the Historical Database and the Complementary Database are generated (Figure 2.2).

The Document Description Database contains the information needed to retrieve the document according to the users' need. This database, plus the source documents, form the basis for a document library.

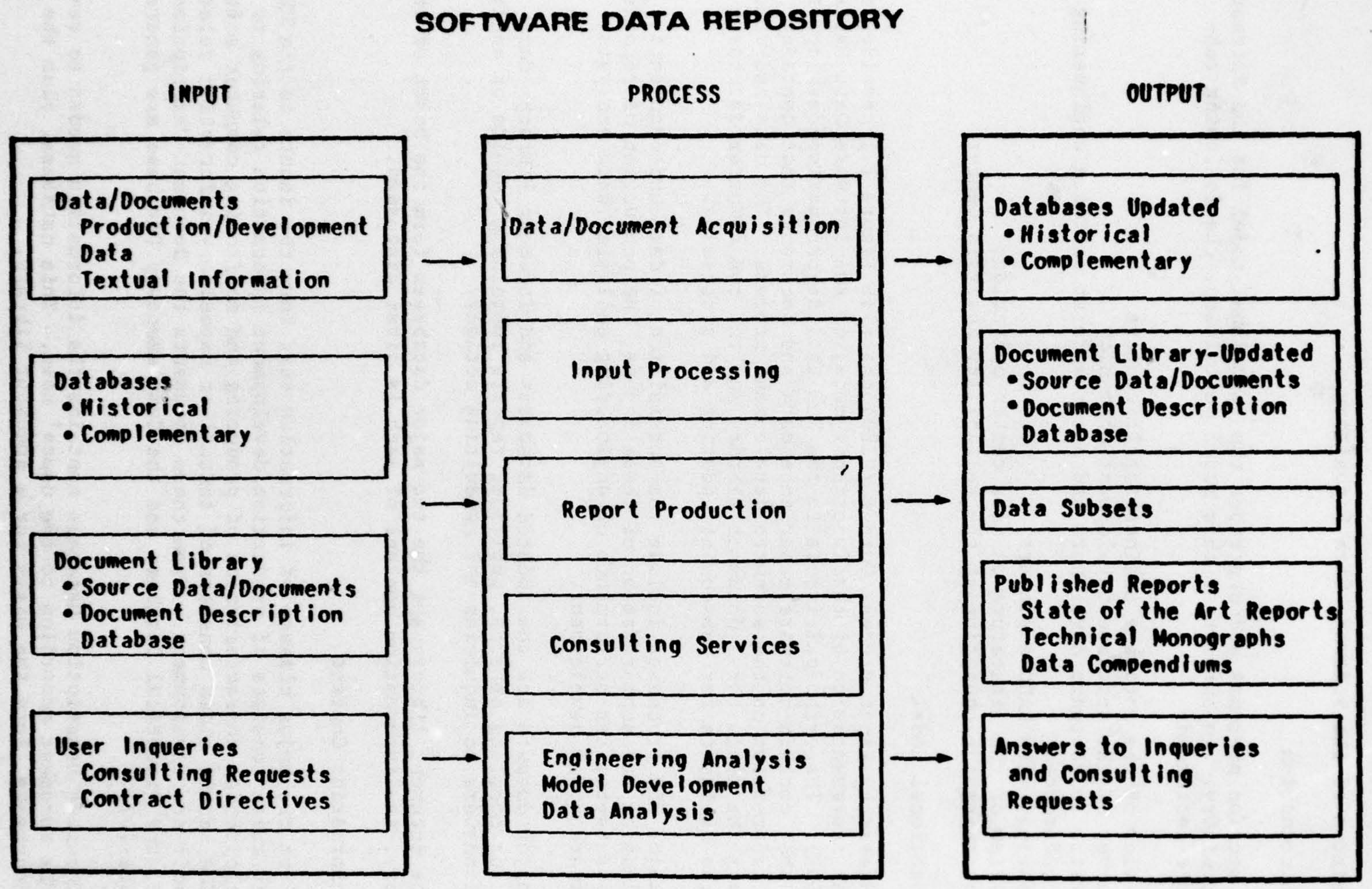


Fig. 2.1 Functional Model

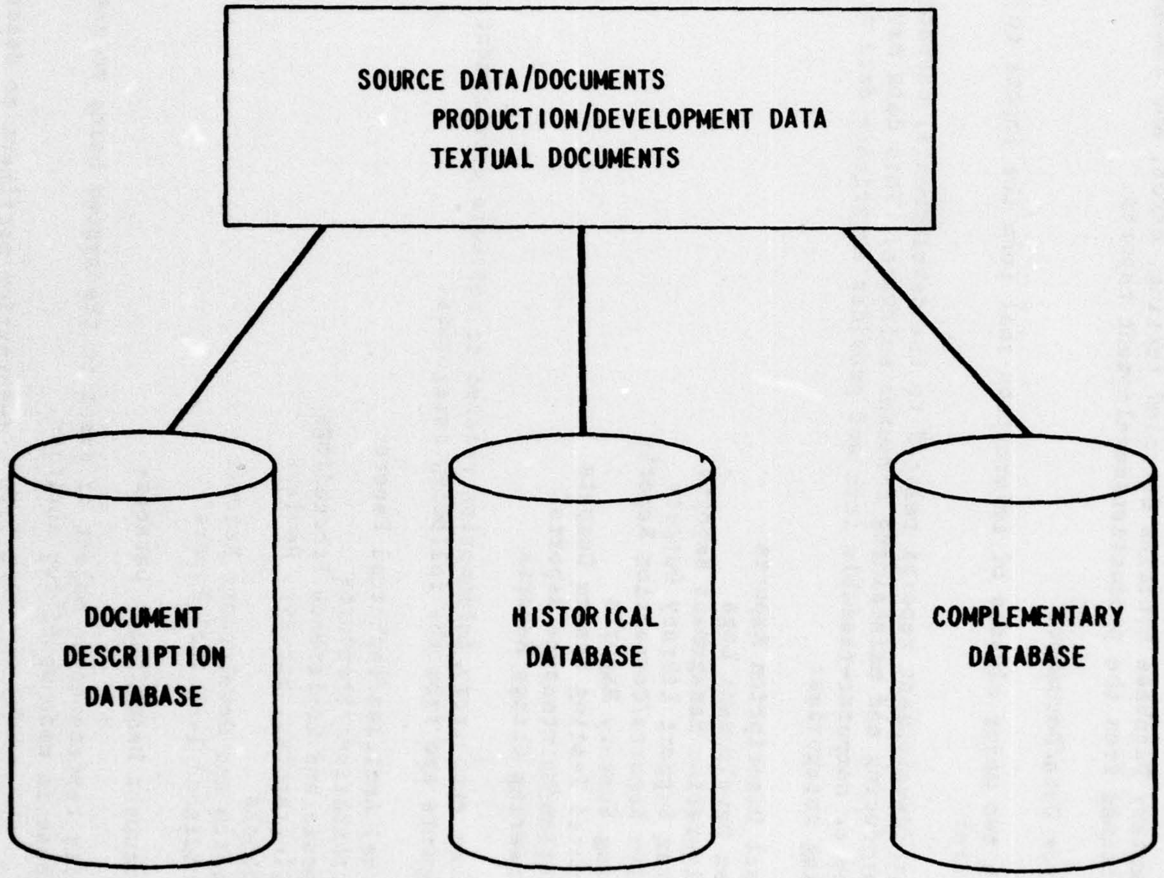


Fig. 2.2 Information Content

The Historical Database contains summarized information obtained from the production/development reports. The types of information contained in this database are productivity, cost, environment, and summarized test and error data.

The Complementary Database contains detailed testing, error, and change information obtained from the production/development reports.

2.3.1 Source Data/Documents

There are two major classes of information that form the inputs to the SDR. They are:

1. Production/development reports related to the developmental characteristics of producing and maintaining computer software. This data may be in hard copy or computer-readable form and provides experience data from the following categories:

- Project Description Reports
- System Development Logs
- Configuration Management Reports
- Program Support Library Outputs
- Problem Reports/Correction Reports
- Testing Summary Reports
- Automated Testing System Outputs
- Operation/Maintenance Reports
- Engineering Change Reports

2. Textual or expository information related to software development. These documents are from the following categories:

- Journal Articles/Technical Papers
- Specifications/Standards
- Symposia and Conference Proceedings
- Bibliographies, Surveys, Reviews
- Handbooks
- Research and Development Reports
- Unpublished Technical Papers

2.3.2 Document Description Database

An entity represents an object or event in the system being modeled. This database is made up of two entities.

- The Document Descriptor Entity - information pertinent to describing the characteristics of the data or the document.

- The Document/Data Cross Reference Entity - information necessary to retrieve the production/development data stored in the other two databases.

2.3.3 Historical Database

The Historical Database contains summarized information from the production/development reports. The data is stored at the project, system and module levels for each phase in the software life cycle.

As used within this report, a <u>project</u> consists of one or more systems and provides a solution to a problem. A <u>system</u> consists of one or more modules and provides a meaningful software product to the user. A <u>module</u> is a discrete identifiable set of instructions handled as a unit by an assembler, compiler or loader.

The software life cycle consists of six phases including the conceptual, requirement, design, implementation, test and operation phases.

1. Conceptual Phase

   Problem statement definition
   Preliminary systems analysis
   Alternative solution categories

2. Requirement Phase

   Project objectives
   System specifications
   Request for Proposal (RFP) preparation
   Proposal preparation and evaluation

3. Design Phase

   Software component definitions
   Interface and data definitions
   Requirement compliance

4. Implementation Phase

   Program code production
   Unit testing

5. Test Phase

   System integration of the software components
   System and acceptance tests
   Requirement compliance

6. Operation Phase

   System use
   System maintenance
      Detection and correction of errors
      Modification to add capabilities and/or improve performance

The Historical Database is made up of six entities (Figure 2.3) and the contents of each is briefly described below.

- The Project/Component Descriptions Entity - descriptive and structural information for the project components; experience information on personnel assigned to the project; and information on the computer system used.

- The Environment Entity - information, for each component and phase, on the types of personnel and computer used, contract type, applicable standards and collection procedures, and project constraints and priorities.

- The Technology Entity - information on the various management, development and testing philosophies, tools and techniques used for each component and phase.

- The Resource Utilization Entity - the calendar time spent, the amount of time for each personnel category and for the computer, and the costs for computer, personnel, travel and miscellaneous items.

- The Production Entity - summarized information, for each component and phase, on the number of pages of documentation produced; the number and types of changes made, tests run, and errors uncovered and corrected.

- The Software Characteristics Entity - the software produced in terms of quality, complexity, stress applied and software constituents.

2.3.4 Complementary Database

The Complementary Database contains detailed information from the production/development reports. As in the Historical Database, the data is stored at the project, system and module levels for each phase in the software life cycle.

The Complementary Database is made up of three entities and is expected to grow as special collections on various projects are made (Figure 2.4).

The first entity is the Project/Component Descriptions Entity as discussed under the Historical Database. If this information is recorded for the Historical Database, it can be integrated into the Complementary Database. The other two entities are the Test Results Entity and the Engineering Change Entity.

- The Test Results Entity - detailed information on the characteristics of each test run including stress applied, error category, severity and correction type, resources used, and affected areas.

- The Engineering Change Entity - detailed information on the characteristics of each engineering change including change type, calendar time and resources used.

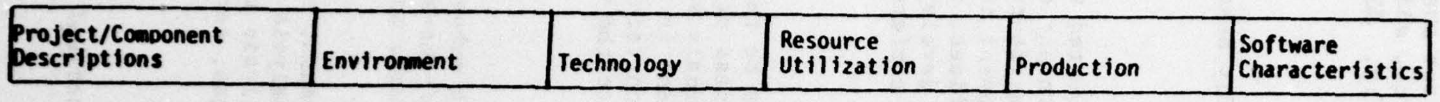


Figure 2.3 Historical Database Entities

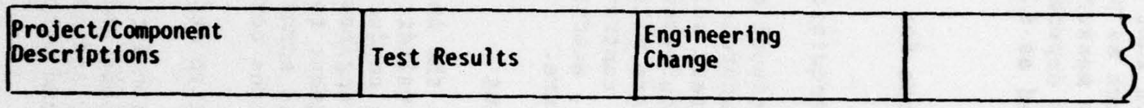


Figure 2.4 Complementary Database Entities

The information in this database is expected to be dynamic to support various research efforts that demand detailed experience data. As special needs are identified such as performing research in error analysis, maintainability, complexity measures, and program constructs, additional entities will be provided dependent upon data availability. Also, obsolete entities will be deleted as data requirements change.

2.4 Process Overview

Major processing elements for the repository are briefly described in the following paragraphs.

2.4.1 Data/Document Acquisition

A concerted and continual acquisition effort is required with full cooperation of the many sources that generate information and data that may be acquired through direct solicitation, voluntary contributions and incorporation of data requirements into software development contracts. The major sources of data are military systems program offices, other government agencies, project contractors, computer software developers, independent R&D organizations, academic community, symposia, conference, seminars, and other open literature.

2.4.2 Input Processing

It is required that the incoming data and documents be thoroughly reviewed for relevancy, validity, accuracy and completeness prior to acceptance. This review is conducted by computer professionals with technical expertise in software engineering. An accepted document is cataloged in the library for subsequent retrieval. If this input contains production/development data, it is summarized and transformed into the SDR formats and then entered into the corresponding database(s).

2.4.3 Report Production and Engineering Analysis

The report production for the SDR implies producing subsets of the data in both computer readable and hard copy form, state-of-the-art reports on designing, developing, and maintaining computer software, and the results of software modeling and other analysis efforts.

The types of analysis include evaluation and refinement of software reliability and maintainability models, regression analysis on the factors that effect productivity and cost, and other data analysis techniques such as distribution fitting, calculation of means, averages, correlations, and frequency distributions.

2.5 Output Types

The major output types and their uses are briefly described in the following paragraphs:

- Database - access to the database for searching and/or copying the contents of the databases.

- Data Subscription - computer readable copy of the database(or a subset of the database) and regular updating service.

- Data compendiums - publications that provide summary information on the database holdings.

- Handbooks - publications that provide practical guidelines and procedures for producing and maintaining computer software.

- State-of-the-Art Reports - publications that provide the latest, authoritative information on software development techniques.

- Technical Monographs - in-depth reports on a specialized software development area such as evaluation and validation of software reliability models.

- Inquery and Consulting Services - these services provide the user with knowledgeable assistance in the solution of a problem.

- Document Library - access to the documents within the library and/or distribution of copies of the documents.

- Bibliography - a publication including references, bibliographic citations, and indices for user search.

3. Input Processing Requirements

3.1 Introduction

One of the most critical and time consuming functions of the repository is that of processing the input to assure that the information is valid, meaningful and easily retrievable. The purpose of this section is to describe the methods required to process the input. Figure 3.1 contains an overall flow of the input processing.

The initial processing of the production/development reports and the textural documents is similar. Both types of inputs are evaluated, indexed, and stored for future reference. If the input includes production/development reports, the data from these reports is transformed and loaded into the Historical and Complementary Databases.

3.2 Evaluation

The first task is to evaluate all incoming data and documents by carefully reviewing the input to determine if the information is valid and is relevant to the software development process. At this point, it is either rejected for inclusion into the SDR or accepted for further processing. Each accepted document is matched against the current source documents for duplication. If a duplication exists, the duplicate document is stored with the original and no further processing is required. If a duplication does not exist, the accepted document is then indexed.

This evaluation process also takes into account the corporate sensitivity of the data or the documents. A determination is made at the time of the levels of security that must be applied to this information.

3.3 Indexing

Indexing is the task of assigning descriptive terms to the input. This is an important process since the access to this information is through the index terms assigned.

The Document Description Form (Figure 3.2) is completed during this indexing task to provide a consistent recording of all indexing information. The indexing information for the reports and documents is grouped into three classes:

1) location term
2) bibliographic terms
3) subject terms or keywords

The location term assigned to any report or document is its accession number. This accession number is unique for each report or document and serves two purposes:

1) it is the primary means of locating the report or document within the library
2) it is the primary means of cross-referencing the report or document for retrieval

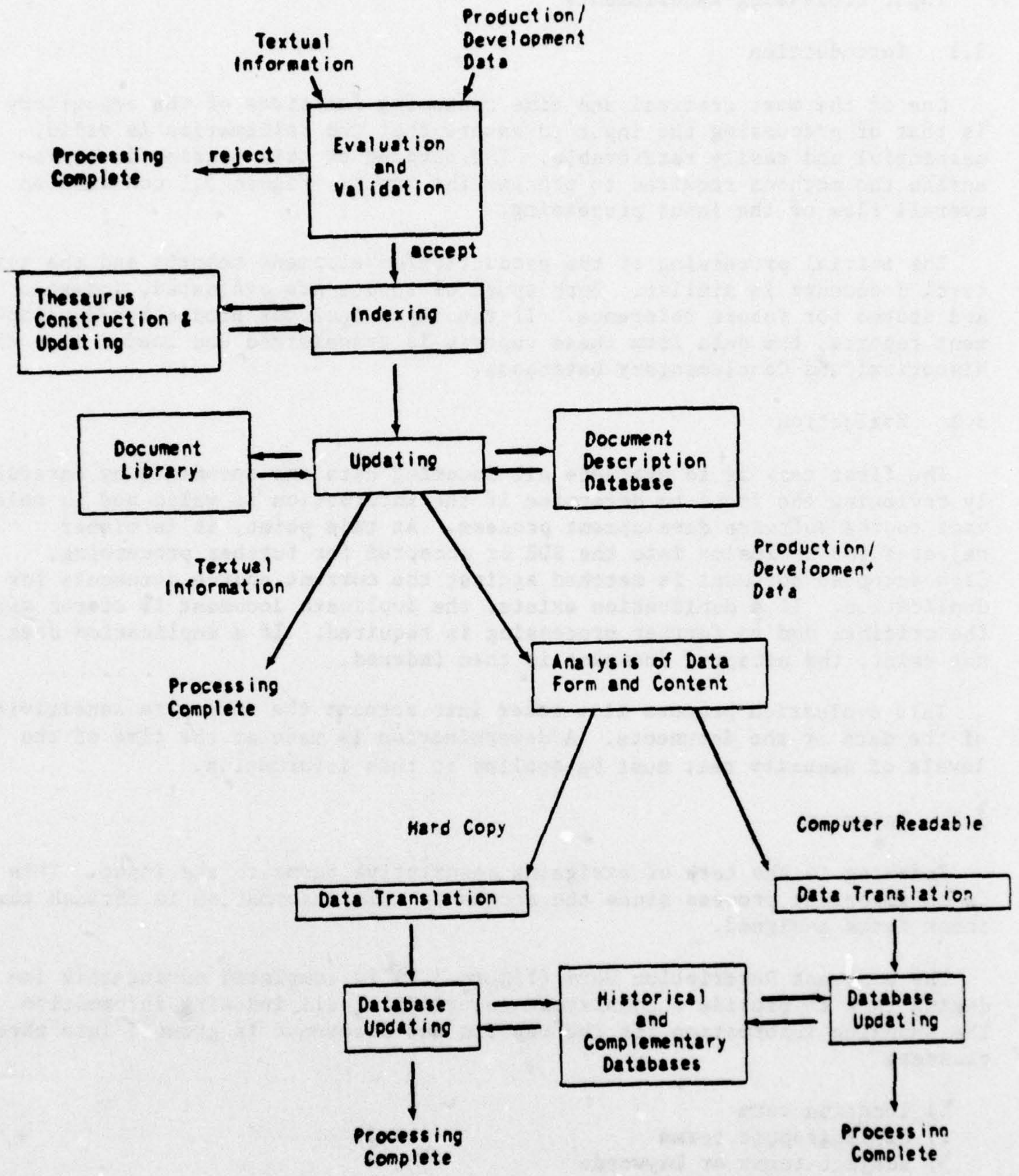
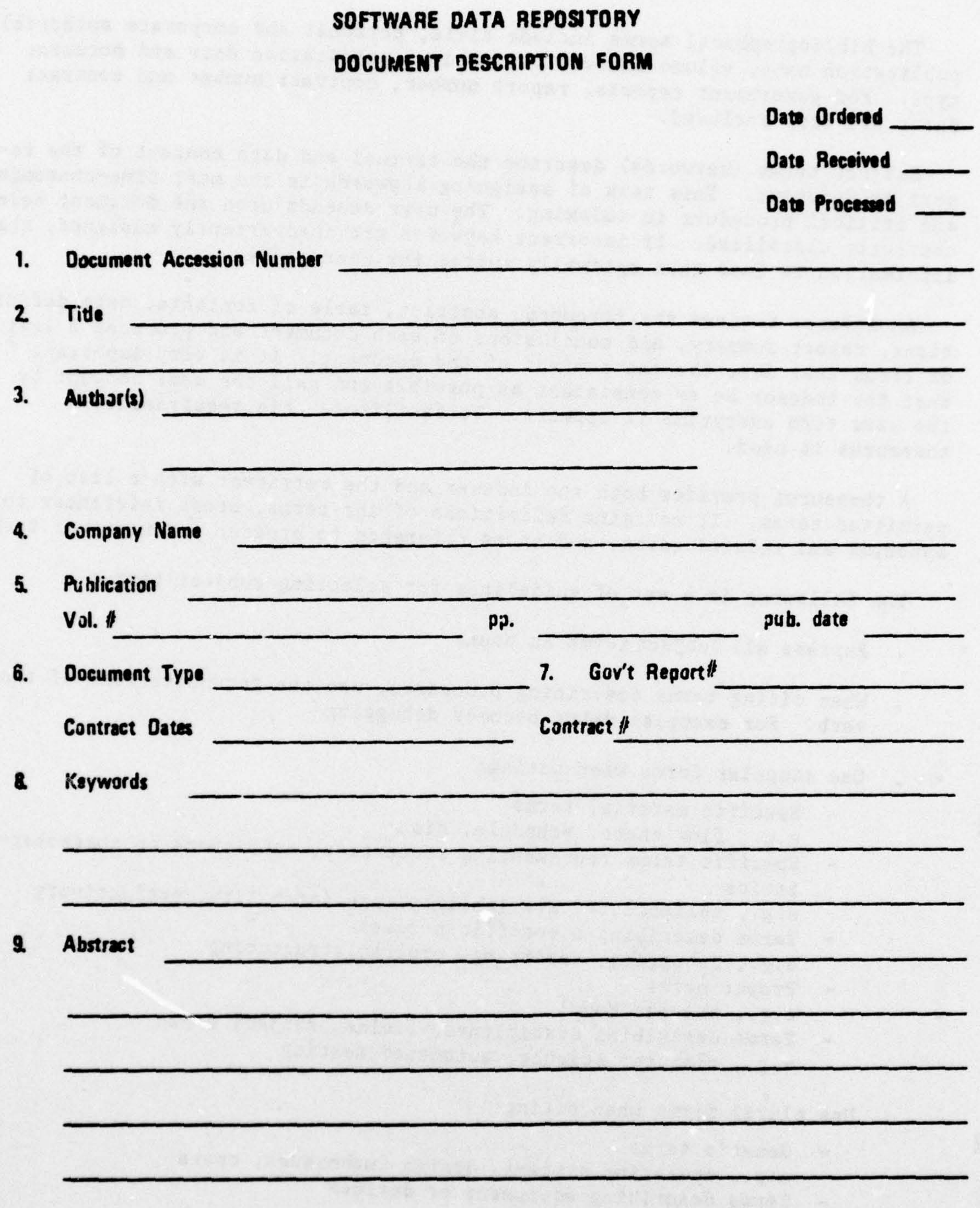


Figure 3.1 Input Processing Flow



SOFTWARE DATA REPOSITORY

DOCUMENT DESCRIPTION FORM

Date Ordered ______

Date Received ______

Date Processed ______

1. Document Accession Number ______

2. Title ______

3. Author(s) ______

4. Company Name ______

5. Publication ______
   Vol. # ______ pp. ______ pub. date ______

6. Document Type ______ 7. Gov't Report# ______

   Contract Dates ______ Contract # ______

8. Keywords ______

9. Abstract ______

Figure 3.2 Document Description Form

The bibliographical terms include title, personal and corporate author(s), publication name, volume number, pagination, publication date and document type. For government reports, report number, contract number and contract dates are also included.

Subject terms (keywords) describe the textual and data content of the report or document. This task of assigning keywords is the most time-consuming and critical procedure in indexing. The user depends upon the document being correctly classified. If incorrect keywords are inadvertently assigned, that information is less than optimally suited for user needs.

An indexer reviews the foreword, abstract, table of contents, data definitions, report summary, and conclusions of each document and produces a list of terms that describe the content of the document. It is very important that the indexer be as consistent as possible and call the same concept by the same term everytime it appears. To facilitate this requirement, a thesaurus is used.

A thesaurus provides both the indexer and the retriever with a list of permitted terms. It contains definitions of the terms, cross references to synonyms and related terms, and cross reference to broader and narrower terms.

The following is a set of guidelines for selecting subject terms:

. Express all subject terms as nouns

. When citing terms describing processes, use the gerundive form of the verb. For example: debug becomes debugging

. Use singular forms when citing:

  - Specific material terms
    e.g., flow chart, schedule, disc
  - Specific terms representing properties, conditions or characteristics
    e.g., reliability, availability, predictability, productivity
  - Terms describing a specific process
    e.g., debugging, verifying, program structuring
  - Proper names
    e.g., Shooman Model
  - Terms describing disciplines, fields, subject areas
    e.g., computer science, automated testing

. Use plural forms when citing

  - Generic terms
    e.g., operating systems, design techniques, costs
  - Terms describing equipment or devices
    e.g., computers, compilers
  - Terms describing events or occurrences
    e.g., failures, interactions

. Use terms that are recognized in the software development community, are unambiguous, and do not occur too frequently in the literature.

3.4 Updating

Updating is the process of storing the source inputs in the document library and entering the information from the Document Description Form into the Document Description Database.

The source inputs are labeled and stored by accession number with a copy of the Document Description Form attached to the document. The particular storage media is dependent upon the document form. Hard copy documents are stored in file folders in filing cabinets, magnetic tapes are stored in tape racks, and punched computer cards are stored in card files.

The information from the Document Description Form is transformed into computer-readable form and entered into the Document Description Database.

The processing for the textual documents is now complete. The production/ development reports require further processing.

3.5 Analysis of Data Form and Content

Since the production/development data is destined for the Historical and Complementary Databases, an analysis is made of the form and content of the data to determine subsequent processing procedures. The data is examined for incompatibilities with the requirements for the Historical and Complementary Databases. These incompatibilities take the form of inconsistent coding values and field lengths, incompatible summarization levels, and distribution of data values.

3.6 Data Translation

The production/development data is transformed to the formats suitable for the SDR databases. This translation is performed using manual procedures and computer programs.

Using manual procedures, this process includes manually transforming the data into the SDR formats and producing a computer readable copy of the transformed data.

Using computer programs, this process includes defining and reading the source data, translating the data according to the SDR data requirements and outputing the transformed data. The extent of the translation is dependent upon the incompatibilities uncovered during the analysis of the data.

3.7 Database Updating

This process includes entering the transformed data into the Historical and Complementary Databases. During the updating process, the data is edited for incorrect coding values, inconsistent records, and duplication of information. If needed, the data is corrected and reloaded into the database.

4. The SDR Information System

4.1 Introduction

An information system for the SDR requires a computer hardware/software system that provides processing capabilities for the SDR and a database management system that provides automated tools for managing the data.

This section presents a general discussion on the requirements of the hardware/software system including the capabilities of the ARPA network, the general requirements of security procedures, a presentation on the requirements of a database management system (DBMS) to manage the SDR databases, and a discussion of the characteristics of the SDR databases.

4.2 Requirements Summary

The basic feature requirements of an information system for the SDR are listed in Table 4.1 Each feature has two columns associated with it. The first column illustrates the need of a feature for a pilot facility, the second column illustrates the need for a fully operational center. The priority need is established according to the following levels:

M Mandatory	This feature must be available.
D Desirable	It would be desirable to have this feature, but if not available, a suitable alternative can be found.
O Optional	It would also be desirable to have this feature but if it is not available, normal operation would not be impaired.
N Not needed	Even if this option were available, it would not be used.
Blank	The requirement level has not been established.

Each of these features are discussed in the following subsections. A glossary of terms is presented in Appendix A for those readers who are not familiar with database terminology.

4.3 Hardware/Software Systems

A hardware/software system to be used by the repository must provide facilities for online and batch processing of standard and ad hoc retrievals, batch processing for standard and any ad hoc reports, online/batch processing for retrievals that result in lengthy answers, batch processing for the maintenance and updating of the database. The system must be able to interface retrieval results and report generation with application programs.

There are at least three options that fulfill these general requirements. The first is to utilize a medium to large scale computer with its attendant software and full range of peripherals, such as found in many computing centers (i.e., RADC computer center). The second is to purchase services from outside vendors. A third option is to purchase a mini-computer and perform all of the processing functions in-house.

Table 4.1

Feature Requirements

Feature	Required	
	Pilot	**Full**
1. Hardware/Software System		
RADC Computer Center	M	
Dedicated Mini-computer	N	
Outside Computer Service	N	
Interface to the ARPANET	O	D
2. Mode of Operation		
Batch Retrieval	O	O
Batch Maintenance	M	M
Online Retrieval	O	M
Online Maintenance	O	O
Online/Batch Retrieval	M	D
Online/Batch Maintenance	D	D
3. Users		
Data Administrator	D	M
Applications Programmer	O	M
Nonprogrammer	M	M
Parametric User	D	M
4. Data Loading and Maintenance		
Self-Contained	M	M
Host Language	D	M
Foreign Files	M	M
Input Verification	D	D
Reorganization	D	M
Restart and Recovery	D	M
Data Translation	D	M
5. Retrieval and Report Generation		
Self-Contained	M	M
Interface to User Progrsms	D	M
Data Subsets	M	M
Multi-Attribute Qualification	M	M
Multi-Attribute Sorting	M	M
Multi-Users	O	M
6. User Programs		
Data Manipulation Language	D	M
Data Files	M	M
7. Database Characterization and Structure		
Data Description Language	M	M
Schema-Subschema	D	M
Data Structuring	M	M
Data Directory	D	M

Table 4.1

Feature Requirements (cont'd)

Feature	Required	
	Pilot	Full
8. Security		
Method		
Administrative	M	M
Hardware	O	D
Operating System	M	M
DBMS	D	M
Technique		
Read Access	M	M
Write Access	M	M
Log-on	M	M
Password	M	M
Level		
Database	D	M
File	D	M
Field	D	M

Any one of these environments could be part of a computer network (such as the ARPA network). A computer network would facilitate direct access to a database and/or the application programs by a multitude of users.

4.3.1 ARPA Network

The ARPA network requirements for the SDR are to provide the facilities for distributing data subsets and receiving foreign files, and to provide direct access to the database by qualified users. RADC has access to the ARPA network through the Multics Operating System.

4.3.1.1 Introduction

The ARPA Network (ARPANET) has been operational since 1969 and is composed of over 85 host computers distributed over 60 different locations in the United States. The geography and topology of the network are illustrated in Figures 4.1 and 4.2 respectively.[307]

The motivations for the network were to establish computer resource sharing, to assist in contractor communication, and to demonstrate the feasibility of packet switching technology.

This network was originally conceived and funded by the Advanced Research Projects Agency (ARPA) of the Department of Defense. The Defense Communication Agency (DCA) has the primary responsibility now and each user is assessed for their network usage. Last year RADC was assessed $60,000. This year it is expected to be $72,000. Bolt Beranek and Newman (BBN) is the primary contractor for the development and operation of the network. The MIT Laboratory of Computer Sciences is presently under contract to develop ARPANET capabilities for the MULTICS operating system on the Honeywell 6180 Computer system.

The major attributes of the ARPANET are listed below.[303]

- A distributed store - and - forward network of heterogeneous computer system (hosts).
- Network Control Programs (NCP) are the software interfaces for connecting hosts.
- The hardware interfaces have the characteristics of either a channel or a communications line.
- An Interface Message Processor (IMP) is a specially modified Honeywell 316 or 516 processor which serves as the communications computer in the network.
- A Pluribus IMP is a multiprocessor IMP assembled by BBN using microprocessors.
- Host to host messages are passed from the host to its IMP, broken into packets, and sent to their destination.
- The route of any given packet is not established in advance and several packets of a message may follow different routes.
- The destination IMP reassembles the message and delivers it to the proper host.
- Three Protocol Levels have been established for communication:

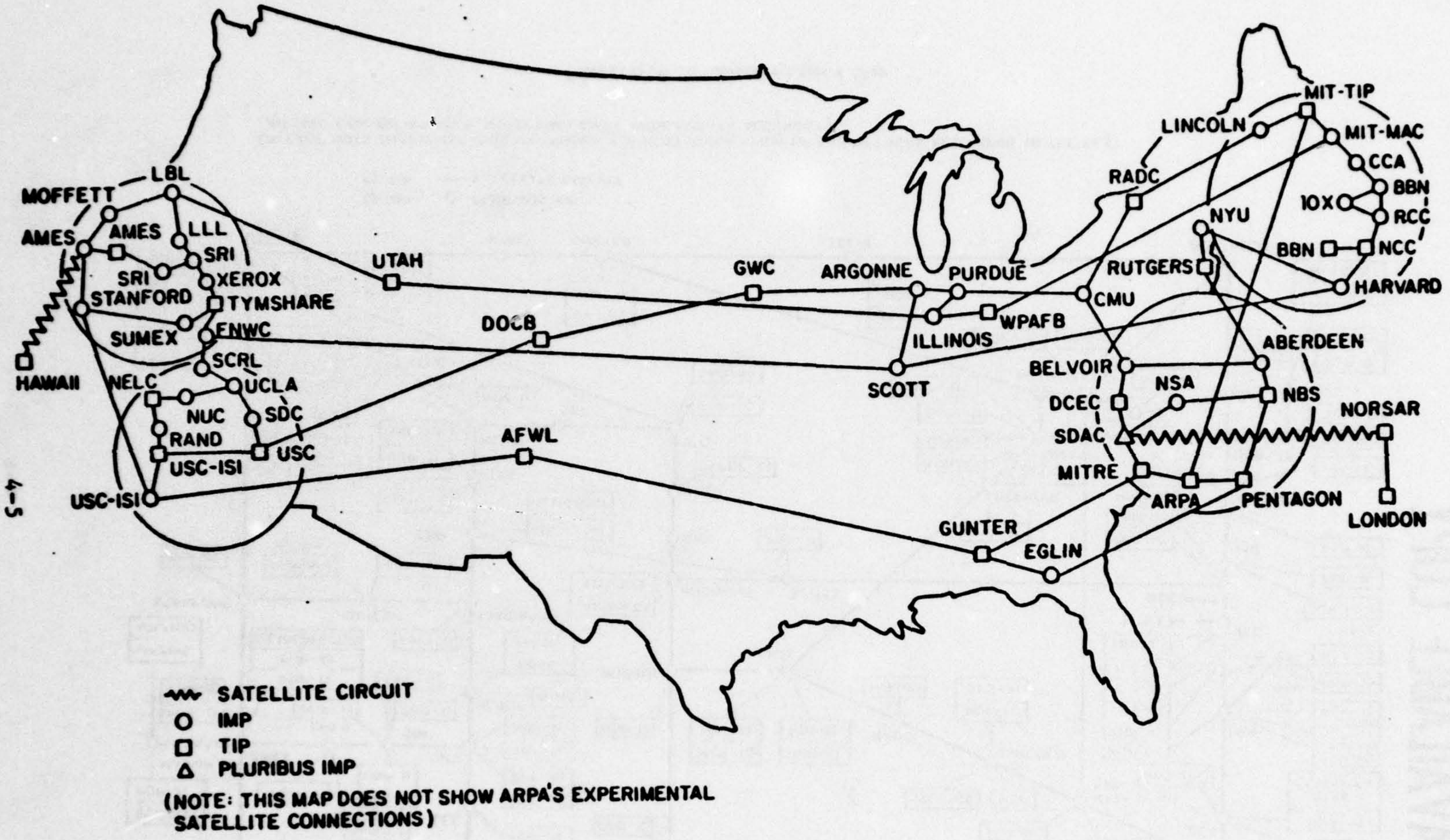
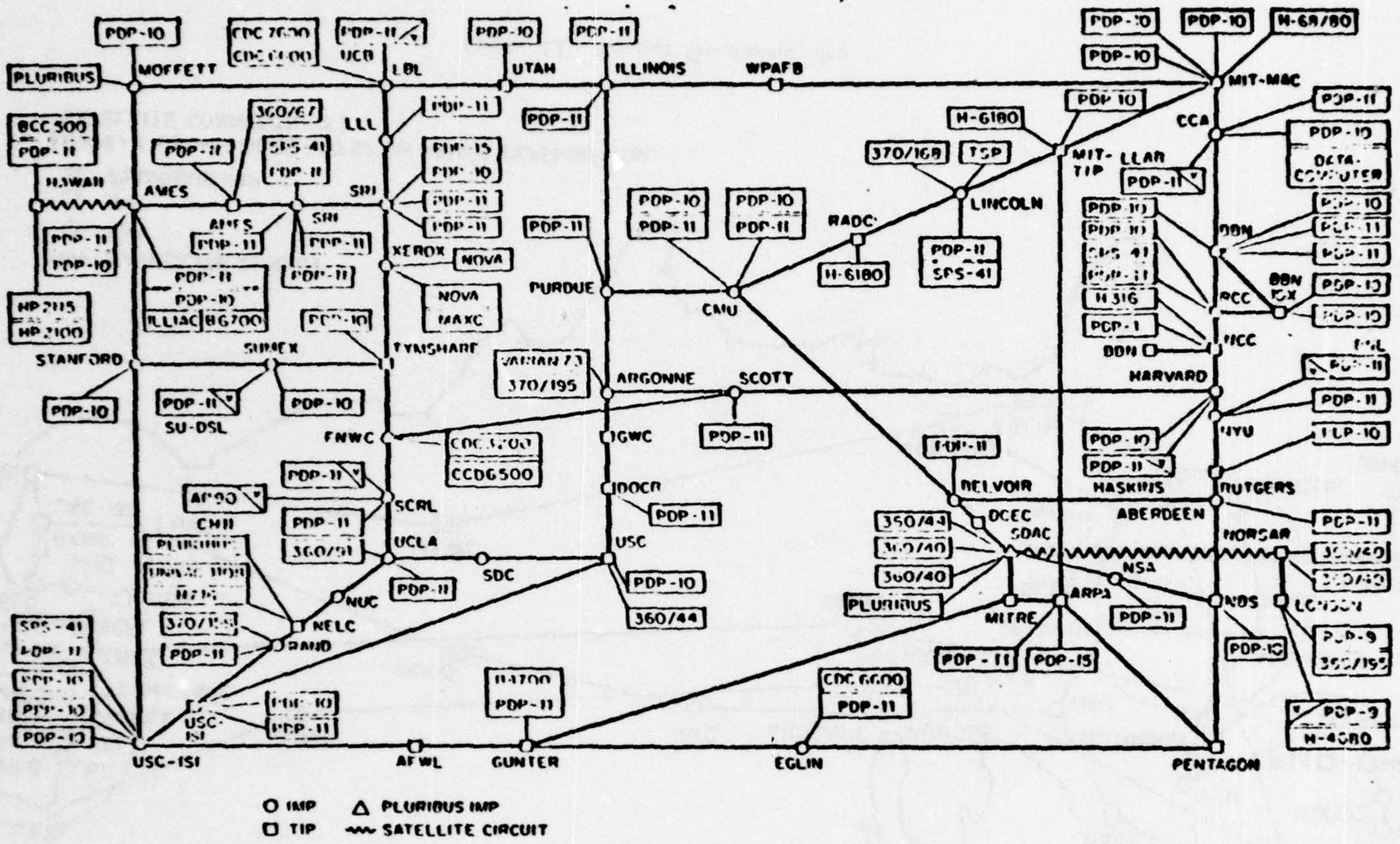


Figure 4.1 Arpanet Geographic Map



(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)

Figure 4.2 Arpanet Logical Map

- Level 1 governs logical exchange of information between host and IMP.
- Level 2 governs logical exchange of information between NCPs in communicating hosts.
- Level 3 governs communications occurring between processes in the host machines. Examples of this level include the data transfer protocol, the file transfer protocol, and the special TELNET Protocol. The TELNET defines a network virtual terminal and permits all terminals on the network to provide a similar interface to processes in a seperate host.

. A Terminal IMP (TIP) is an IMP which is augmented by additional memory and a multiline controller. The TIP contains a network control program and a TELNET program to permit terminals to access the network directly

4.3.1.2. RADC - ARPANET Accessibility

RADC is presently a TIP user on the ARPANET through the Multics Operating System (Figure 4.3). Below is a brief summary of the status of ARPANET accessibility by RADC:

. The lower level protocols (Levels 1 and 2) are well established and would be of little concern to the SDR.
. The file transfer protocol (FTP) is also well established. The transfer parameters include type (ASCII, BCD, etc.), mode (stream, block), and structure (file, record). This protocol is used extensively for transferring messages (mail facility). In addition to this protocol, there is a need for data transformation software (as discussed in a previous section).
. The other higher level protocols are not as well established and the user must learn the host dependent commands and conventions (log-on procedures, job control language, file naming conventions, etc.). In addition, the quality of documentation and assistance availability varies from site to site. There is a need for a system control language to provide a common means for users and application programs to access the resources needed, independent of their network location.[092]
. The communication system is highly reliable.
. The reliability of the host system varies from site to site.
. Backup processing capability is left to the user.
. File security is the responsibility of the individual host.

4.3.2 SDR Options for Utilizing the ARPANET

The ARPANET can be used for the repository at RADC to transfer subsets of the production/development data to qualified users, and/or to receive production/development data from the software developer.

A higher level of usage of the ARPANET is to allow qualified users a retrieval capability to provide them with access to the Document Description, Historical and Complementary Databases. In addition the repository personnel may retrieve production/development data from the developers' database to determine applicability to the repository before actually receiving the data.

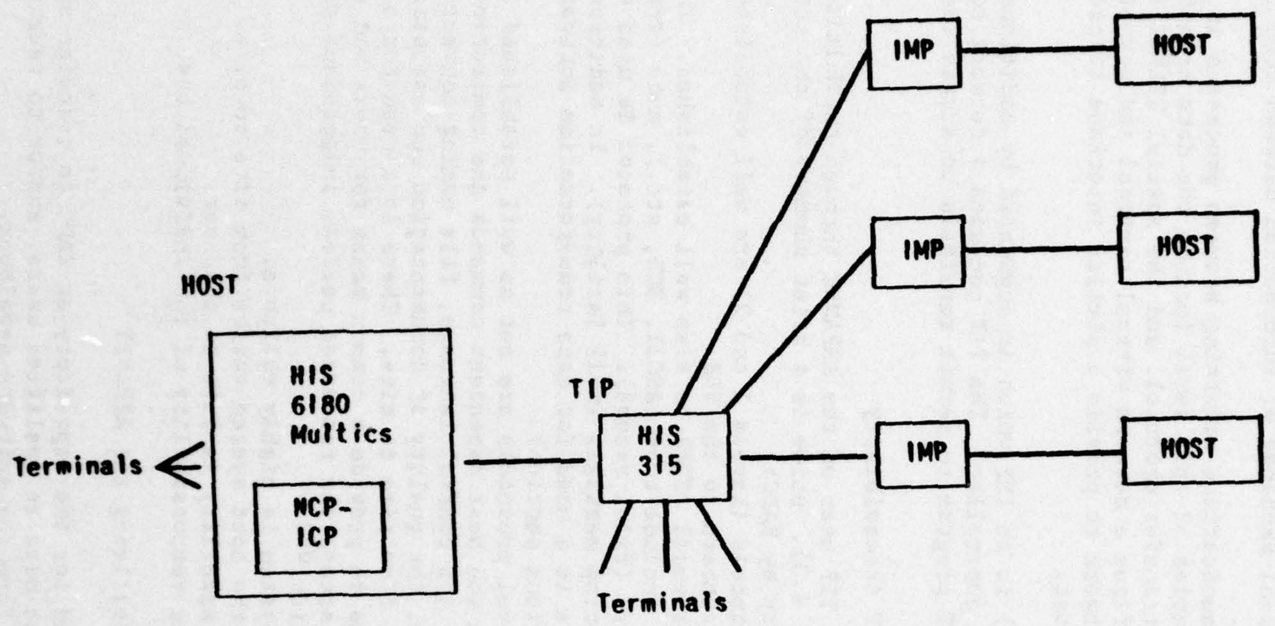


Figure 4.3 RADC - ARPANET Configuration

4.4 Security

As some of the product/development reports will include corporate sensitive productivity and cost data, procedures and mechanisms must be established and utilized to prevent unauthorized access to this sensitive data. In addition, there must be security provisions to safeguard against the accidental distruction of data and the disruption of user services.

The SDR security involves the computer system including its attendent hardware and software components, the repository users, the repository and computer site personnel, the physical environment, and the relationships and interactions between all of the above (Figure 4.4).

Backup procedures such as restart and recovery can be incorporated to provide some level of data integrity. Also editing and validation can be accomplished to prevent the invalid entry of data.

Adequate technological safeguards to provide protection from deliberate misuse or access to the system are not now available in current computer systems.[305] The use of passwords for access to the operating system and the database are available but can be penetrated. These mechanisms will be used, but, at least initially, the sensitive data submitted to the SDR will physically be stored for manual retrieval rather than to make it available through the computer system.

Sensitive data is that data which has been submitted to the repository that the submitter considers sensitive. That is, the judgement of sensitivity will be left up to the contributor. It is expected that the sensitive data may consist of the developers' affiliation, project designations, productivity data, and cost data.

4.5 Database Management System Requirements

Shown in Figure 4.5 are the major database management system functions that must be considered for the SDR.[101] Each of these functions is discussed below including a discussion of the types of users of a database management system.

4.5.1 Users

The Conference on Data Systems Language Systems Committee (CODASYL)[206] defines four levels of users of database management systems. These are data administrator, applications programmer, nonprogrammer and the parametric user. The data administrator is the one person who is responsible for the creation, restructuring, security, etc. of the database.

The next level is the applications programmer. These persons are computer professionals who are well versed in the current practices of data processing. They are responsible for writing application programs, processing difficult queries, generating reports, etc.

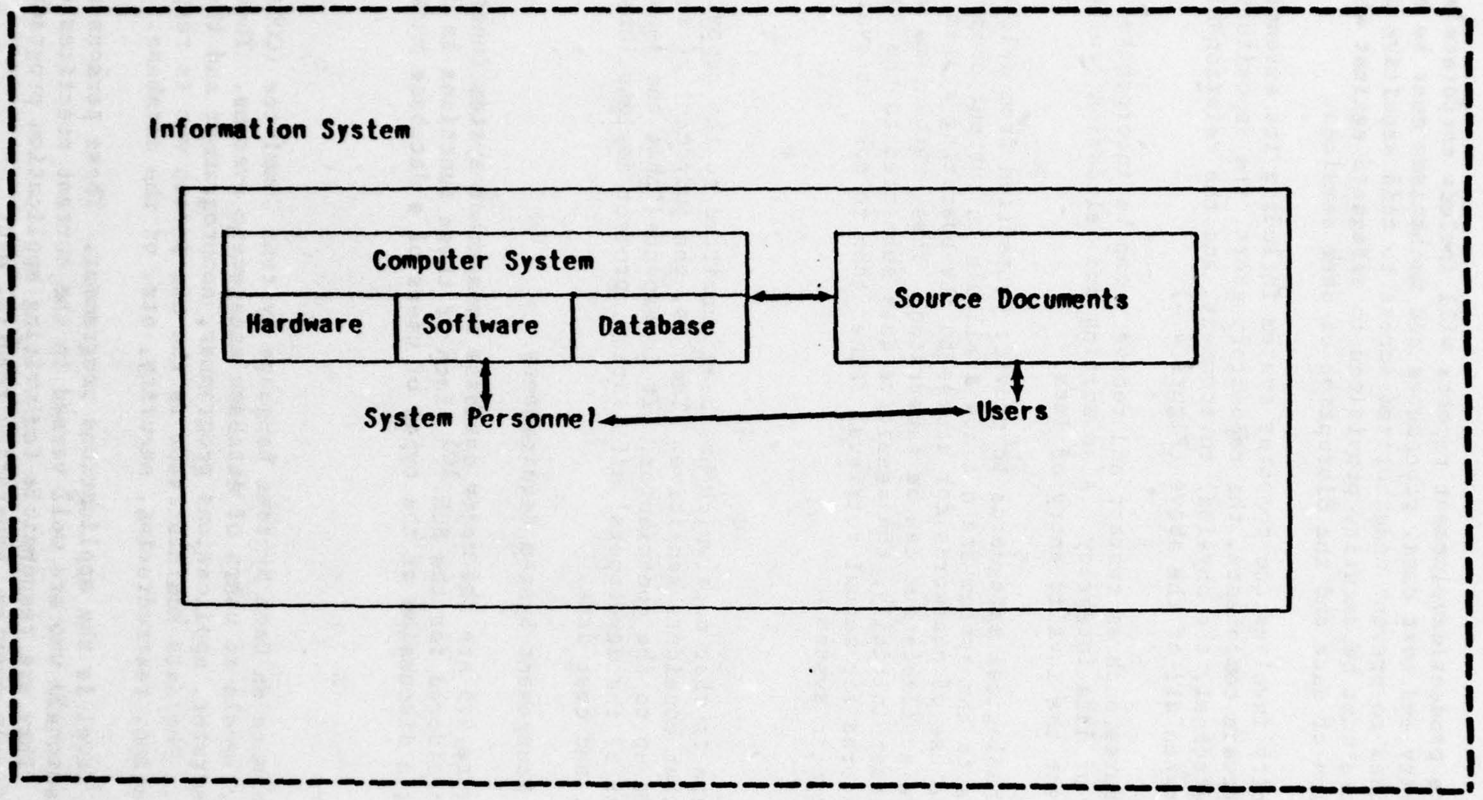


Figure 4.4 Security Elements

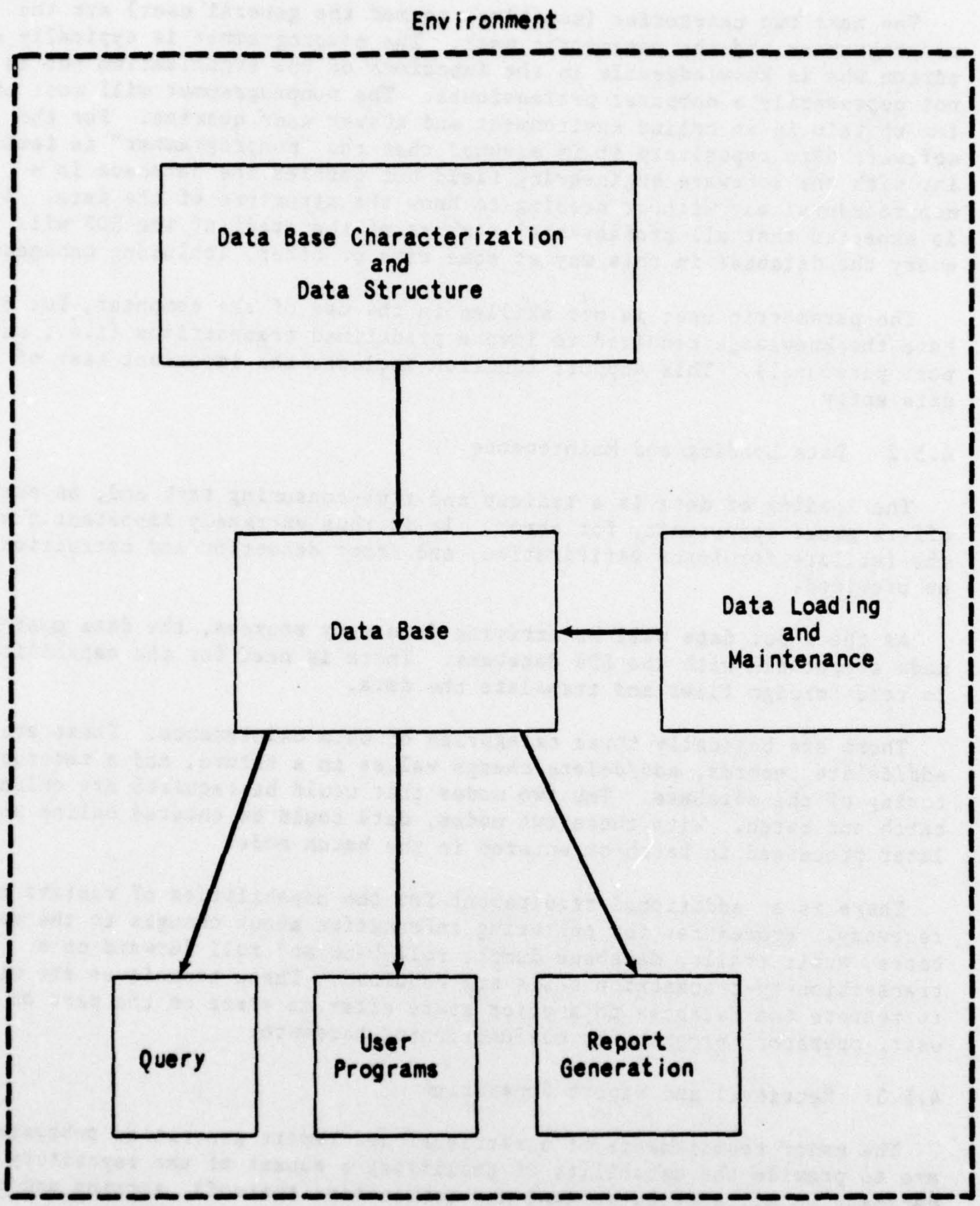

Environment
Data Base Characterization and Data Structure
Data Base
Data Loading and Maintenance
Query
User Programs
Report Generation

Figure 4.5 Database Management System Functions

The next two categories (sometimes termed the general user) are the nonprogrammer and the parametric user. The nonprogrammer is typically a person who is knowledgeable in the functions of the organization but is not necessarily a computer professional. The nonprogrammer will most often operate in an online environment and answer user queries. For the software data repository it is assumed that the "nonprogrammer" is familiar with the software engineering field but queries the database in a nonprocedural way without needing to know the structure of the data. It is expected that all professional members of the staff of the SDR will query the database in this way at some time or other, including management.

The parametric user is not skilled in the use of the computer, but does have the knowledge required to invoke predefined transactions (i.e., support personnel). This support function includes the important task of data entry.

4.5.2 Data Loading and Maintenance

The loading of data is a tedious and time-consuming task and, as such, offers great opportunity for error. It is thus extremely important that the facility for input verification, and error detection and correction be provided.

As the input data will be arriving from many sources, the data must be made compatible with the SDR database. There is need for the capabilities to read foreign files and translate the data.

There are basically three categories of data maintenance. These are add/delete records, add/delete/change values in a record, and a restructuring of the database. The two modes that would be required are online/batch and batch. With these two modes, data could be entered online and later processed in batch or entered in the batch mode.

There is an additional requirement for the capabilities of restart and recovery. Procedures for capturing information about changes to the databases, audit trails, database dumps, roll back and roll forward on a transaction-by-transaction basis are required. These techniques are used to restore the database to a prior state after an error on the part of the user, operator, program, or malfunctioning hardware.

4.5.3 Retrieval and Report Generation

The major requirements of a retrieval and report generation subsystem are to provide the capability of qualifying a subset of the repository database on all attributes (and any combination thereof), sorting and/or formatting this subset, interfacing this subset to user programs, or printing this subset directly to the requesting computer terminal. Since some of the users of this system will be nonprogrammers, there is a requirement for online tutorial material to aid in retrieval specifications. It would also be highly desireable to have feedback on long retrievals. This would require a combination of online and batch. Thus, retrieval specification would take place online and the actual processing would be performed in batch. There might also be the need for some standard retrievals which would be run quite often and, thus, would be predefined.

It is anticipated that the repository would produce reports summarizing the data. Therefore, this subsystem would have to have the capabilities for multi-attribute qualification, running several different user programs against the database, specifying output columns and headings, placing several different values under each heading, and placing limited free-form remarks under certain headings.

There is the need for the generation of ad hoc reports which may occur as a result of an online batch retrieval or may originate as a batch job.

4.5.4 User Programs

In order to support the repository adequately, the system will have to provide for the running of a wide variety of user programs. These programs might be written in FORTRAN, COBOL, JOVIAL, PL-1 or some other standard programming language, and require the use of a data manipulation language for reading, writing, searching, and updating the database. Depending upon the function of the program, it must be able to operate online, in batch and/or in the online/batch mode.

The user programs will vary from major modeling systems to statistical analysis routines acting on subsets of the database. Some examples of the programs that would be required for statistical analysis include mean, range, variance, standard deviation, confidence intervals, F-test, T-test, chi-square test, analysis of variance, regression analysis, analysis of covariance and probability distribution fitting. Programs of nonstatistical nature would include sort-merge, qualification to obtain a subset of the database, and various routines that would be required for the generation of reports.

4.5.5 Database Characterization and Structure

A database management system must have the capability to define, load and update the data, retrieve on the attributes of the data, produce reports and be able to interface with user programs. There must be a data description language to define the data structure, attribute type and length, and to handle coded values within the value sets.

As discussed in Section 2, there are three SDR databases:

. Document Description Database
. Historical Database
. Complementary Database

The Document Description Database contains descriptive information on the documents within the document collection. This database is made up of two entities, seven groups, and 17 attributes (Figure 4.6).

As defined before, an entity represents an object or event concerning the software development process. An attribute describes the entity and is the smallest named logical unit of data within the database. A group is a named collection of attributes.

DOCUMENT DESCRIPTOR ENTITY

Group Name		Attribute Names
Title, Author	Accession #	Title, Personal Author, Corporate Author
Citation	Accession #	Type of document, Name, Volume number, Year
Report	Accession #	Report number, Contract number
Keywords	Accession #	Keywords
Abstract	Accession #	Abstract
Processing	Accession #	Ordered date, Received date, Indexed date, Entry date

DOCUMENT/DATA CROSS REFERENCE ENTITY

Group Name		Attribute Names
Document/ Data Structure	Accession #	Summary ID#

Figure 4.6 Document Description Database

The Historical Database contains summarized environment, resource utilization, production, testing, and change data. This database consists of six entities, 33 groups, 107 attributes and 33 value sets (Figure 4.7). A value set is the set of all possible values for an attribute. The initial values for 24 value sets are illustrated in Figure 4.8. The values for the additional nine value sets will be determined as data is accepted into the database.

The Complementary Database contains detailed testing and change data. This database consists of three entities, 14 groups, 67 attributes and 20 value sets (Figure 4.9).

Illustrated in Figure 4.10 are the entities and groups of the three databases shown in a hierarchical manner.

4.5.5.1 Data Independence

As we learn more about the software development process and the factors that affect cost, productivity and quality, the types of data that are collected and included in the databases will change. As this will be an evolving process rather than a sudden change, there is a need to reflect these modifications minimizing the effects on the operation of existing programs. There is a need to provide for independence from the physical storage device, the ability to add new fields to the records, and the ability to alter relationships. One method of minimizing the effects of these changes is through the concept of schema and subschema, and of execution binding.

The schema is a description of the database (Figure 4.11). The subschema is a description of the data that is needed to meet the requirements of a particular program. The schema is translated by a database management system into a set of stored schema tables. This is done separately from any application program. At program compilation time, the subschema is also translated into a set of tables. Then at execution time, the database management system "binds" these two tables together in order to provide the data as described in the subschema, to the program. The conventional approach is to "bind" the data to the program at compilation time rather than execution time.

PROJECT/COMPONENT DESCRIPTIONS ENTITY

Group Name		Attribute Names
Project	Project ID	Name, Description, Project Type[1], # Systems, # Modules
System	System ID	Name, Description, System Type[1], # Modules
Module	Module ID	Name, Description, Module Type[1]
Component ID#-P	ID#	Project ID, System ID, Summary ID#
Component ID#-S	ID#	System ID, Module ID, Summary ID#
Job-Personal	Person ID	Job experience[1], # Years
Application-Personal	Person ID	Application experience[1], # Years
Rating-Personal	Person ID	Rating Criteria[2], Rating measure[2]
Computer Hardware	Computer ID	Classification[2], Device type[2], Manufacturers name[2], Model number, Number of devices, Storage capacity
Computer Software	Computer ID	Software type[1]

[1] Value Sets are Available in Figure 4.8

[2] Value Sets are data dependent

Figure 4.7 Historical Database

ENVIRONMENT ENTITY

Group Name	Attribute Names		
Person	ID#	Phase[1]	Person ID, Person level[1], % Time, Travel required
Computer	ID#	Phase[1]	Computer ID, % Time, Operating Model[1], Response[1], Reliability[1]
General	ID#	Phase[1]	Contract Type[1], Standards[1], Collection Purpose[1], Collection Procedures[1], Priorities[1]

TECHNOLOGY ENTITY

Group Name	Attribute Names		
Technology	ID#	Phase[1]	Tech. ID[1], Tool Name[2], Cost of Usage

RESOURCE UTILIZATION ENTITY

Group Name	Attribute Names		
Calendar Time	ID#	Phase[1]	Estimated begin date, Estimated end date, Actual begin date, Actual end date
Person Time	ID#	Phase[1]	Person level[1], Estimated # person days, Actual # person days
Person Cost	ID#	Phase[1]	Person level[1], Estimated person cost, Actual person cost
Computer Time	ID#	Phase[1]	Usage level[1], Estimated usage level, Actual usage level
Computer Cost	ID#	Phase[1]	Estimated computer cost, Actual computer cost
Travel	ID#	Phase[1]	Estimated # trips, Actual # trips
Travel Cost	ID#	Phase[1]	Estimated travel cost, Actual travel cost
Misc. Cost	ID#	Phase[1]	Estimated misc. cost, Actual misc. cost
Total Cost	ID#	Phase[1]	Estimated total cost, Actual total cost

1 Value Sets are Available in Figure 4.8

2 Value Sets are Data Dependent

Figure 4.7 Historical Database (cont'd)

PRODUCTION ENTITY

Group Name	Attribute Names		
Documentation	ID#	Phase[1]	Document Type[1], # Pages, Reporting period
Instructions	ID#	Phase[1]	Instruction Type[1], # Instructions, Reporting period
Changes	ID#	Phase[1]	Change Type[1], # Changes, Reporting period
Test	ID#	Phase[1]	Test Type[1], # Tests, # Failures, Reporting period
Error	ID#	Phase[1]	Error Category Type[2], Error Category[2], # Errors, Reporting period
Corrections	ID#	Phase[1]	Correction Type[2], # Corrections, Reporting period

SOFTWARE CHARACTERISTICS ENTITY

Group Name	Attribute Names		
Quality	ID#	Phase[1]	Quality measure type'- Major[1], Quality measure type-Minor[1], Quality measure, Method [2]
Complexity	ID#	Phase[1]	Complexity measure type[1], Complexity measure
Testing	ID#	Phase[1]	Stress type[1], Stress measure
Constituents	ID#	Phase[1]	Constituent type[1], Constituent name, # Of occurrances

[1] Value Sets are Available in Figure 4.9
[2] Value Sets are Data Dependent

Figure 4.7 Historical Database (cont'd)

Application Experience

- Programming language
- Computer system
- Application area

Person Level

- Manager
- Systems Analyst
- Senior Programmer
- Junior Programmer
- Computer Operator
- Librarian
- Clerical
- Other
- All

Software Type

- Operating system
- Telecommunication system
- Data base management system
- Utilities
- Application programs

Phase

- Conceptual
- Requirements
- Design
- Implementation
- Test
- Operation

Operating Mode

- Batch
- Remote batch
- Time-sharing
- Time-sharing/batch

Response

- Average turn-around time

Contract Type

PRIM	- Prime contractor
SUB	- Sub-contractor
CO	- Co-contractor
INT	- Internal project
CPFF	- Cost Plus Fixed Fee
CPIF	- Cost Plus Incentive Fee
COST	- Cost with no fee
T/M	- Time and Material
FFP	- Firm Fixed Price
FPIF	- Fixed Price Incentive Fee

Standards / Specifications

Air Force
- AFSR 310-1(TD-3)
- MIL-R-R-83313/11521
- MIL-STD-483(USAF)
- SAFSL EXHBT.20012
- AFSCP 375-1
- AFR 65-3
- AFSCP-800-3,6
- SAMSO EXHIBIT 73-3

DOD
- DOD DIRECT.5010.19
- DOD 4120.17-M
- MIL-STD-490
- MIL-STD-100A
- MIL-STD-480, 481, 482

Navy
- NAVORD WS 8506
- SECNAV INSTR. 5233.1
- NAVMATINST 4130.1
- NAVAIRINST 4130.1

Army
- AR 700-51
- AMCR 11-26

NASA
- NHB 80.40.2
- SHUTTLE CM

Internal

Collection Purpose

- Contract
- Visibility
- Evaluate personnel
- Evaluate approaches
- Schedule monitoring
- Cost monitoring
- Reliability monitoring

Collection Procedures

- Configuration Management
- Software Implementation Monitor (SIMON)
- Management Data Collection and Reporting System
- Integrated Management, Project Analysis and Control Technique (IMPACT)
- Internal
- Other

Constraints

- Limited development computer accessibility
- Operational computer different than development computer
- Unstable design
- Unstable requirements
- Unrealistic schedule
- Inadequate funding
- Inadequate staffing
- Limited management support
- New application area
- Hardware limitations

Figure 4.8 Value Sets

Priorities

- Processing speed
- Core utilization
- Schedule
- Cost
- Reliability
- Maintainability
- Transferability

Usage Level

- CPU
- I/O
- Algorithm

Document Type (DOD 4120.17M)

- Functional Description
- Data Requirements Document
- System/Subsystem Specification
- Program Specification
- Data Base Specification
- Program Maintenance Manual
- Users Manual
- Computer Operation Manual
- Test and Implementation Plan
- Test Analysis Report

Instruction Type

- New
- Modified
- Throw-away
- Re-used

Change Type

- Hardware modification
- Efficiency
- Modified requirements

Test Type

- Functional test - selective
- Functional test - exhaustive
- Logical test - selective
- Logical test - exhaustive
- Proof of program correctness

Quality Measure Type - Major

- Completeness
- Conciseness
- Understandability
- Portability
- Consistency
- Maintainability
- Testability
- Usability
- Reliability
- Structuredness
- Efficiency

Quality Measure Type - Minor

- Reliability
  - # Errors/1000 lines code
  - Probability of success
    (designate model utilized)
  - Failure rate
    (designate model utilized)

Complexity Measure Type

- Number of instructions
  (excluding comments)
- Halstead length
- C2 measure
- F2 measure
- M measure
- D measure

Stress Type

- Units of CPU time
- Wall clock time
- Number of instructions executed - total
- Number of instructions executed - unique
- Number of test executions

Constituent Type

- Assembly language instructions
- HOL instructions
- Object words
- Unique variables
- User macros
- System macros
- Comment statements
- Mathematical instructions
- Internal control instructions
- External control instructions
- Input/Output instructions
- Unique input formats
- Unique output formats
- Average # fields/input-output format
- Data items in data base
- Pages of documentation

Test Result

- Success
- Failure

Figure 4.8 Value Sets (cont'd)

TECH ID

Chief programmer team
Configuration management
Model driven software

Automated analysis
Automated design tools
Automated requirements tools
Bottom-up design
Decision tables
Design language
HIPO design aid
Process design language
State diagrams
Structure charts
Top-down design

Modular decomposition
Program support library
Simulation
Structured programming
Walk-throughs

Assembly language
Bottom-up programming
Comparator
Critical piece first
Data base analyzer
Data dictionary
Documentation generator
Editor
Higher order language
Pre-compiler
Program restructuring
Re-usable code
Top-down programming

Assertion proofs
Bottom-up testing
Code standards auditor
Compatability checker
Data base comparator
Independent test team
Program flow analyzer
Program structure checker
Reliability modeling and measuring
Test case generation
Test data generation
Top-down testing

Figure 4.8 Value Sets (cont'd)

TEST RESULTS ENTITY

Group Name	Attribute Names		
Test Info	ID#	Phase [1]	Test ID, Test Type[1], Date run, Time run, Stress type[1], Stress measure, Test result[1], Problem report number, Report date
Error Info	ID#	Phase [1]	Problem report number, Error category type[2], Error category[2], Origin[1], Severity type[2], Severity[2]
Correction	ID#	Phase [1]	Problem report number, Correction type[2], Date began, Date ended, Computer resources used, Person resources used, # Instructions affected, # Pgs. documentation affected

ENGINEERING CHANGE ENTITY

Group Name	Attribute Names		
Change	ID#	Phase [1]	Change type[2], Change number, Date initiated, Date approved, Date completed, Person resources, Computer resources, Total cost, Change ID#

[1] Value Sets are Available in Figure 4.8

[2] Value Sets are Data Dependent

Figure 4.9 Complementary Database

BEST AVAILABLE COPY

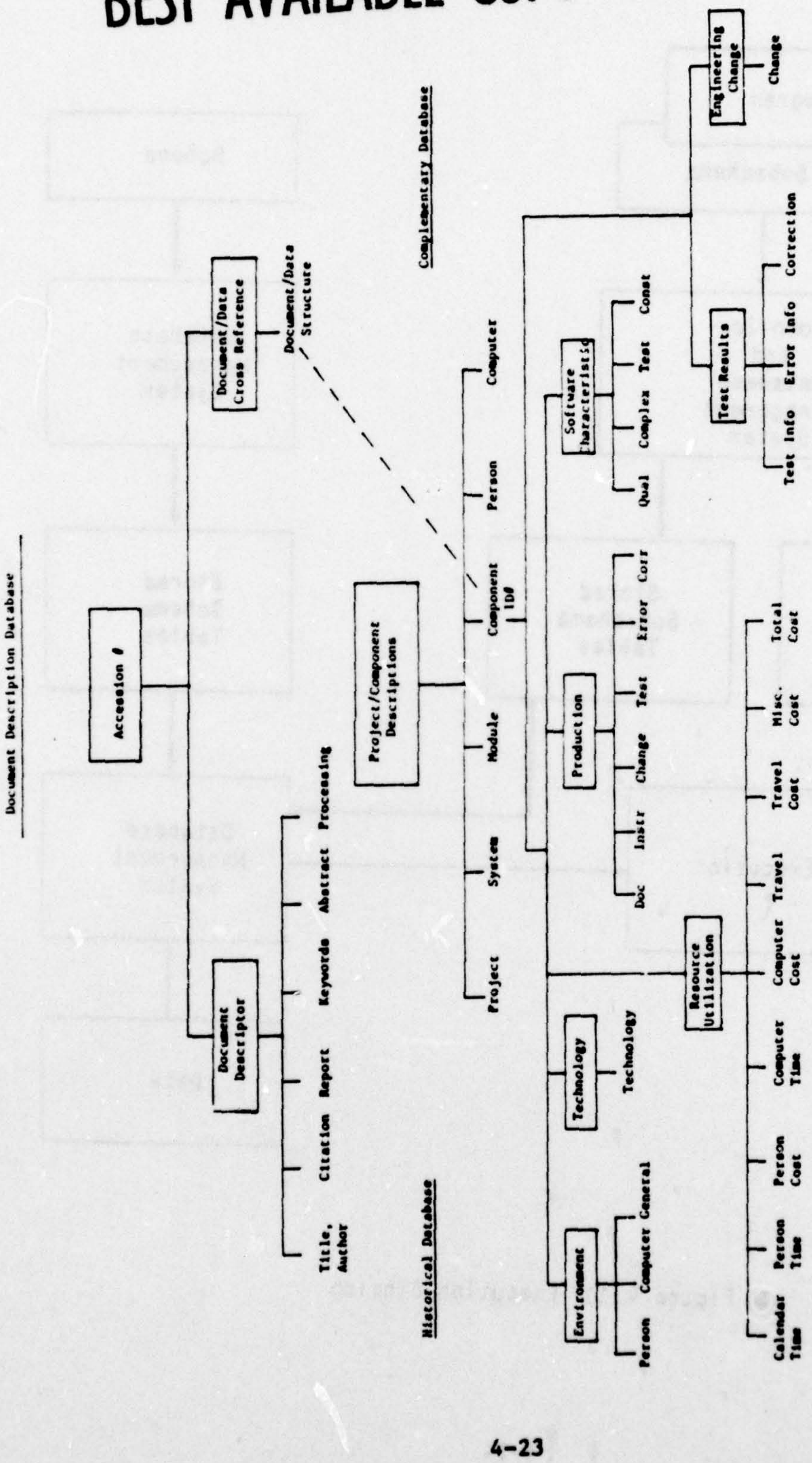


Figure 4.10 SDR Databases - Hierarchical Representation

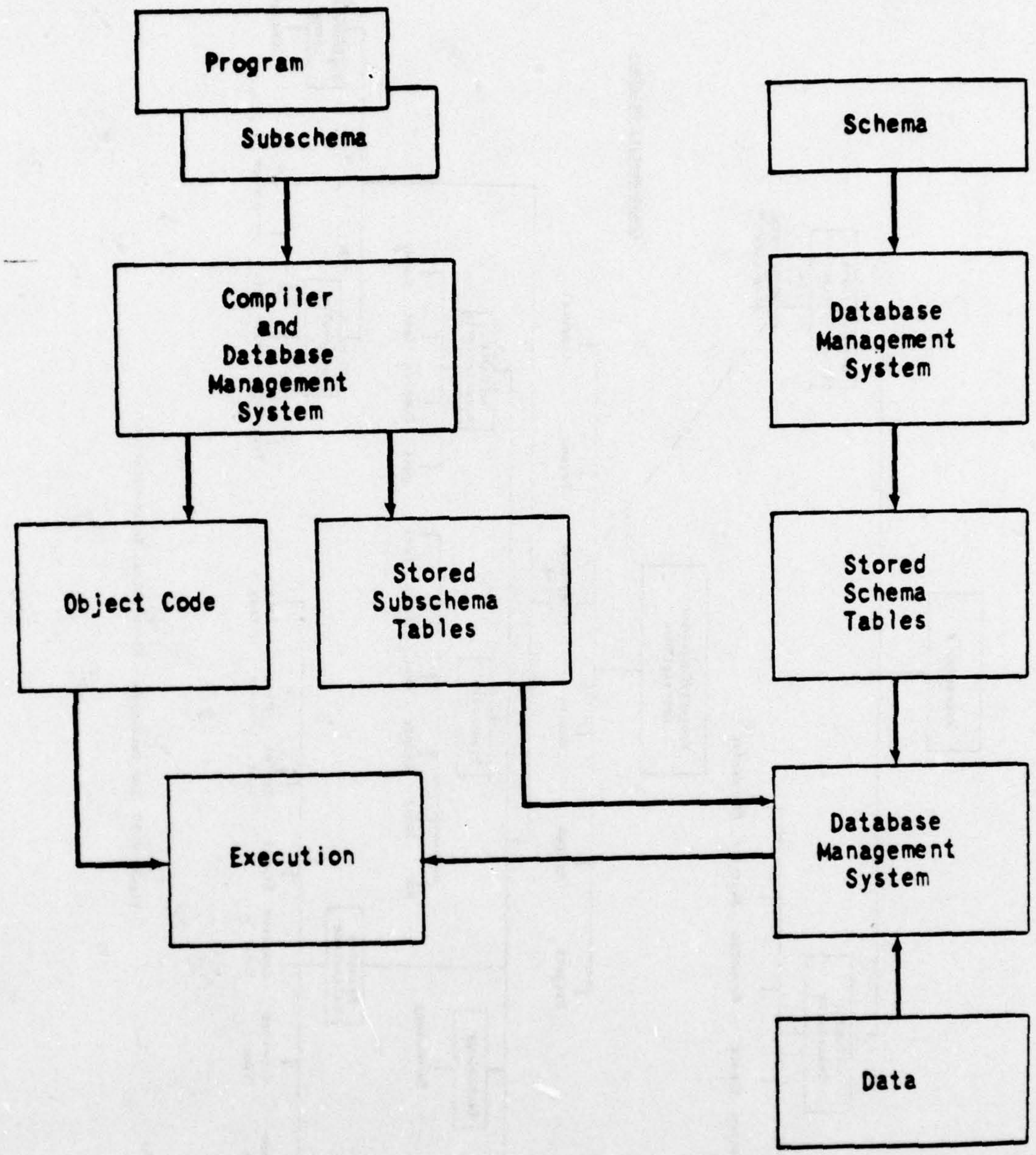

Program
Subschema
Compiler and Database Management System
Object Code
Stored Subschema Tables
Execution
Schema
Database Management System
Stored Schema Tables
Database Management System
Data

Figure 4.11 Execution Binding

5. System Considerations

5.1 Introduction

This section contains discussions on:

1. systems that automatically and manually collect software data,
2. data entry hardware devices that are available on the market for consideration as tools to transform hard copy data to computer-readable form,
3. alternative information system and database management system approaches, and
4. alternative approaches to document library automation

The reader who is only interested in the recommendations for the pilot facility would not lose the flavor of the repository if this section was skipped. It is expected that this section will be used more for reference purposes.

5.2 Data Collection Systems

The purpose of this subsection is to provide a general description and data applicability to the repository of seven representative systems that collect software experience data. The purpose of these systems vary and are described for each case. Table 5.1 contains an indication of the types of data each system provides.

5.2.1 Management Data Collection and Reporting System

The Management Data Collection and Reporting System (MDC&R) is being developed by the IBM Federal Systems Division under contract to RADC. It was initially specified in the Structured Programming Series Reports and is part of the Program Support Library (PSL).[043, 079, 087] The purposes of this collection system are to provide management with information for project control and to improve our understanding of the software development process. They have identified 112 data items for manual and automatic collection.

They divide the data into two major classes, plan data and actual data. The plan data describes the characteristics of the project as originally planned, or as better estimates become available; the actual data describes the characteristics of the project as the development progresses. Each of these two major classes is further divided into five types: project environment, module, computer utilization, resource cost and program production.

Before explaining these five types, it is important to delineate their five levels for collecting data. The unit level is the lowest level for collecting data. The unit is a named subdivision of a program which can be treated as a single entity. The second level is the program. This is the lowest level that can be assembled (or compiled) and executed as

Table 5.1

Data Collection Systems

Entity/Group Name	System Name						
	MDC&R	SIMON	PET	ASES	IMPACT	RXVP	JAVS
Project/Component Descriptions							
Project	3	3					
System	2	2	1				
Module	2	2	1	1	2	2	2
Component ID	3	3			3		
Person	3	1			3		
Computer	2				1		
Environment							
Person	3	1			3		
Computer	2				2		
General	1	1			1		
Technology							
Technology	2						
Resource Utilization							
Calendar Time	3	2			3		
Person Time	3	2			3		
Person Cost	3				2		
Computer Time	3	3			3		
Computer Cost	3	3			3		
Travel	3				1		
Travel Cost	3				2		
Misc. Cost	3				3		
Total Cost	3	3			3		
Production							
Documentation	2				2		
Instructions	2				2		
Change	2				3		
Test	2		2		3		
Error	2	1			3		
Corrections	3				3		
Software Characteristics							
Quality							
Complexity	2	2	1		2	2	
Testing					2	2	2
Constituents	2	2	1	1	1	3	3
Test Results (Detailed)							
Test Information			2	1	3	1	2
Error Information					3		
Correction					3		
Engineering Change							
Change					3		

Key: 3 - Provides the majority of the data required
2 - Provides an adequate amount of the data
1 - Provides a mimimal amount of data
Blank - No data is provided

a single entity. The third level is the computer job, where a job is made up of one or more job steps. A job step can be a compilation, utility run, etc. The fourth level is the subsystem, which is comprised of one or more programs, and represents the computer software that is developed for a project.

The project environment data provides a general description of the project and of the software system to be developed; and characteristics of the personnel and the customer. The module data describes the unit and the program in terms of dates, programming language, and name of programmer. The source code also constitutes part of the module data.

The computer utilization data describes the use of the computer, by job, in terms of computer turn around time. The resource cost data describes the costs, by subsystem, for materials, personnel, travel and computer.

The program production data describes the development of the programs and units in terms of number of compilations and changes; number of update executions; and the number and types of improvements made or errors corrected.

As illustrated in Table 5.1, the Management Data Collection and Reporting System provides a good portion of the kinds of data required for the SDR databases. It is lacking in its collection of detailed testing, error, and correction information but does provide some summarized testing and error data. Data on component descriptions, constraints and priorities, quality, and engineering changes are not available through the system but could be provided as supplemental information.

5.2.2 Software Implementation Monitor

The Software Implementation Monitor (SIMON) is being developed by Mitre Corporation under contract to RADC.[021-023, 031, 085] The major purposes of this system are to provide for the systematic and consistent collection of data for research into factors affecting software quality and cost, and to provide technical and managerial visibility of the software development process.

They have defined 111 data items that should be collected during the development cycle. The data is divided into five major types: project, person, subsystem, module, and error-disc-information.

The project data describes the project as a whole; the person data describes the assignments of programmers to subsystems; the subsystem data describes the status of each subsystem in terms of resources used (persons, computer, etc.) and phases complete; the module data describes the program or unit of compilation in terms of complexity measures, program attributes, and status; and the error-disc-information describes the error history during development including number and types of discrepancies and errors by subsystem and total.

SIMON offers a viable approach to provide data to the SDR but it lacks in its collection of detailed testing, error and correction information although some summarized error data is provided. Data on component descriptions, personnel experience, constraints and priorities, technology and production are not available and only computer and total cost data is provided. All of the above, except for production data, could be provided as supplemental information.

5.2.3 Program Evaluation and Tester System

The Program Evaluation and Tester (PET) System is a test evaluation tool developed by McDonnell Douglas which automatically generates self-metric or self-measuring software from existing software.[082,083] Self-measuring refers to the ability to measure the thoroughness of testing to which a piece of software has been subjected by analyzing sensors that have been inserted into the software at selected points. Summary and detailed reports are produced that quantify the degree of testing to which a program has been subjected.

The types of data collected relate to the paths in a program that have or have not been tested. Although this system does not collect any cost or productivity data, it could be used in conjunction with other data collection systems to provide information on software validity.

5.2.4 Automated Software Evaluation System

The Automated Software Evaluation System (ASES)[060-063] was originally designed by the University of California at Berkeley under the sponsorship of the U.S. Army Safeguard System Evaluation Agency to assist in the development of large scale software. The system produces reports that provide a list of statements, cross reference listings, language constructs, warning messages and structural characteristics of the program. It provides a static analysis by producing lists of undefined labels, unreferenced labels, unreachable statements, statements with no successors within the program, and direct predecessors to all labelled statements. It provides for error detection related to semantics, language constructs and program structure.

The software data collected is similar to that generated by a compiler; therefore, the data collection capability is very limited. This limitation, together with the fact that the system was designed to operate for a specific programming system (CENTRAN) make it undesirable as a data collection tool for the SDR.

5.2.5 The Integrated Mangement, Project Analysis and Control Technique

The Integrated Management, Project Analysis and Control Technique (IMPACT) is a component of the Software Factory developed by System Development Corporation.[011-013, 099] The purpose of IMPACT is to assist the manager in evaluating project progress and spot potential difficulties and development trends.

IMPACT provides for the storage of schedule, resource allocation, personnel assignment, programs, requirements and milestone information. The data files for IMPACT are created and maintained by utilizing data coded onto forms called Task Milestone Worksheets. These worksheets comprise a variety of forms all of which are required to completely describe a project.

The first of the series of worksheets is called the IMPACT Information Item Definition Worksheet. This worksheet is used to record program module descriptions, milestone events, documents to be referenced or written, modifications to be processed and validation criteria for testing and specification reviews. The Module Task List Worksheet is used to record schedules, work units, resource allocation and responsibility assignment data. Also entered on this form is cross-reference information pointing to software documentation, milestones, module files, design change records and software tests and validation criteria.

The third Task Milestone Worksheet is called the Module Environment List. This form is used to record module, data and equipment interdependencies. This data is used to establish the hierarchial structure of the software system. The Task Configuration Worksheet is used to record tasks, subtasks, task interdependencies and formulas for estimating resource requirements per unit of work for the task. The Milestone Event and Software Certification Authentication Log is used to record milestone events, software test schedules, milestone completion dates and the name of the person who certified the milestone as having passed. This information is used to report software configuration status. The Modification Milestone Log is the sixth Task Milestone Worksheet. This form is used to record the processing steps to be followed for proposing, evaluation, approving, and implementing changes.

The last Task Milestone Worksheet is used to record personnel assignments. This data includes the employee name, person number, skill category, organization code, and for each work assignment, the module name, task name and charge number. The time expended on each task is added to the information when a task is completed.

In addition to the information mentioned above as manual input to the IMPACT data base, several production data items are captured automatically from compiler generated information. These data items include:

1. Number of compilations per module
2. Number of tests executed
3. Computer time expended on compilations or test
4. Module size (number of statements)
5. Cross reference information
6. Number of errors encountered

The combination of the manual and automatically collected data constitutes the information retained in the IMPACT database. The IMPACT database comprises the major portion of the project control data and is the major source of information used by software development managers to monitor project activity.

IMPACT provides a large percentage of data required for the SDR databases. It is lacking the project and system descriptions including constraints, priorities and technologies used. However, these could be collected independently of the IMPACT System. In addition, more detailed configuration management data is collected than is required of the repository and therefore summarization is needed.

5.2.6 RXVP Automated Verification System

The RXVP System was developed by General Research Corporation for further research aimed toward identifying and solving problems of software reliability and quality and evaluating automated tools which enhance software reliability.[050-052, 058] RXVP collects static and dynamic program behavior data by the insertion of software "probes" or "sensors" at each decision point in the Fortran program so the outcome of each decision can be recorded.

All of the data collected by RXVP is collected automatically. The data retained in the database is related to test results and module behavior during testing. The types of module data captured include:

- name, number, type
- number of statements, symbols, entries, paths, external modules referenced, paths (2 types), executable statement, branch executions
- complexity measurements, symbol usage checks
- essential paths whose execution is necessary to ensure complete testing coverage.

The RXVP data could form the basis of an entity within the Complementary Database for the calculation of complexity and quality measurements. However, because it is Fortran limited and there is no cost, productivity or project description data, this would be of limited utility unless these kinds of data had already been provided through other data collection systems.

5.2.7 Jovial Automated Verification System

The Jovial Automated Verification System (JAVS) was developed by General Research Corporation under contract to RADC and is a similar system to RXVP operating on JOVIAL source code.[016, 017] JAVS analyzes the JOVIAL source code, assists the user in preparing test cases, and analyzes the results of tests.

As with the RXVP data, the JAVS data could be used in conjunction with supplemental cost and productivity data to perform studies on complexity and quality.

5.3 Data Entry Hardware Options

The data entry function is defined as those operations necessary to transform required information from its original form to a form most suitable for computer processing. This subsection presents a discussion of alternative hardware devices that assist in this transformation process.

The traditional mode for achieving the data entry function involves the presentation of the source data on an input document which is then read by an appropriately trained and equipped operator and transformed to computer-readable form (such as punched cards or tape, or magnetic tape or disk, etc.) by the entering of the information via some type of keyboard. The keyboard, device itself may be used as a stand-alone unit (i.e., keyboard, control and output are all self-contained), may be online to a host computer for interactive entry/response, or may be one of a multi-station installation (i.e., more than one keystation under the control of a single processor and centralized output storage).

5.3.1 Keyboard Devices

The keypunch has been around for many years and will continue to play a significant role in the future in data entry procedures. However, new technological advances of the past few years allowing the relatively inexpensive replacement of hard-wired controller logic with state-of-the-art microprocessors and semi-conductor memories, coupled with cost reductions which have been effected for magnetic storage drives, now allow a number of other keyboard-to-storage approaches to be seriously considered as very viable alternatives.

For the relatively low volume card-oriented environment a buffered keypunch should be considered. At approximately $170 per month, little more than the combined cost of an unbuffered keypunch and verifier, the capabilities of both can be combined into one unit, with the added benefit of significantly faster operation and normally many more allowable program formats.

Another alternative, occasionally justified, is the keyboard to punched tape device typified by the Teletype ASR 33, where data is entered onto punched tape in an offline mode and then transmitted over communications networks to a central computer.

A variety of stand-alone keyboard-to-magnetic storage devices may be used to store data onto computer-compatible magnetic tape, magnetic tape cassette or cartridge, or fixed or removable disk or diskettes. For these devices the stored data is conveyed to the central computer by the pooling onto a magnetic tape or disk and/or transmitting the consolidated information via data lines. These devices offer significant advantage over keypunch approaches in that record sizes are no longer limited to the 80- (or 96-) character constraint of a punched card, upper and lower case alphabetics can be input to the system, the storage medium can be reused, and throughput is dramatically increased.

Stand-along key/magnetic tape keystations cost on the order of $150 to $250 per month. However, at least one station must be equipped with a pooling and/or data communications capability which will add another $250 to $300 per month.

Because of their added reliability and simplicity of use, the key to cassette, cartridge, diskette, and disk units are currently in much higher demand than the key/magnetic tape devices. Key to disk devices start at about $400 per month and the others at about $150 to $250 per month. As with the key to magnetic tape devices, however, at least one unit must have a pooling and/or data communications capability which may add another $200 or $300 per month.

Keyboard entry can also be effected through online interaction with a host computer. If this approach is used, online editing of the data as it is entered can be performed. However, there is increased operator-associated overhead resulting from timesharing response delays coupled with the added communications line costs if the central computer is not onsite.

If online data entry is elected, either hard- or soft-copy (e.g., CRT display) terminals can be employed for the data entry activities. Hard-copy terminals, in general, cost less than their soft-copy counterparts and offer the distinct advantage of a hard-copy record of all transactions. The soft-copy units, on the other hand, are generally more reliable and quiet, and besides not everyone needs a hard-copy record of all transactions from the terminal. Costs for hard-copy devices can range from under $50 per month for the 10 character per second Teletype KSR 33 to about $150 per month for a typical 30 character per second device. For about $175 to $200 per month plus a modem, a 120 cps device may be acquired.

Soft-copy units are primarily CRT displays but could also incorporate other techniques such as LEDs or plasma displays, or even an audible signal. Typical limited capability display devices can be acquired for about $100 per month while those with more desirable features are generally available for about $150 per month.

In many cases, the same hard- and soft-copy devices available as online data terminals are also available as integral parts of the keystations used in many key/storage devices.

For installations with substantial data entry volumes, multi-station key to storage devices should be given serious consideration. Such units are considerably more costly than stand-alone devices and require significantly more planning and preparation for their proper use. However, under the proper circumstances, the vastly enhanced operator productivity and system capabilities in the form of relaxed program format and block size restrictions and improved editing, verification, and validation capabilities, can make it all more than worthwhile.

Being a complete computer processing facility in miniature, multi-station key-entry systems come at a premium price, with typical costs running from about $1,000 per month to an eight-station installation with minimal capabilities to about $4,000 to $5,000 per month for more elaborate capabilities and additional stations. The cost per station for the average eight key installation generally runs between $130 and $200.

5.3.2 Nonkeyboard Data Entry Procedures

In many instances, particularly where there is a high volume of data and there are trained operators in a controlled environment, the elimination of the keyboard entry step through the use of mark or character recording can be most effective. For such instances where the amount of information recorded on a single document is small, the use of mark recording may be very much in order. If the information requirements are such that the data entry document can be designed to be accommodated through the use of pencil marks in selected areas of the document, this technique could indeed by the best alternative.

If the mark sense form can be incorporated onto a punched card, costs for the required mark scanner are relatively low - typically running at about $250 per month. Because of the added complexity of the document feeding mechanism, scanners for other sizes of marked documents are somewhat more expensive, typically running in excess of $1,000 per month. Such mark readers can be used offline, recording onto magnetic storage, usually tape, or else directly online to a remote job entry (RJE) terminal or a host computer.

Where remotely-generated information must be processed in very high volume, the use of machine character recognition techniques may be in order. While devices do exist for directly processing handwritten characters, their high cost coupled with the prevailing high rejection rates for such documents except for very strictly constrained circumstances, precludes their serious consideration at this time.

On the other hand, many successful applications have been affected with the use of automatic character recognition techniques using highly stylized character fonts. A number of such fonts have been standardized in the industry and there are a variety of page and document readers available to directly generate the required digitized information from such documents. Typical optical character scanners for the stylized fonts can be installed for upwards from $1,500 per month while those recognizing limited handprinted information usually cost in excess of about $2,500 per month.

Another viable option for large volume installations is the use of a so-called mixed media system which combines an optical character recognition (OCR) or optical mark recognition (OMR) reader with a multi-station key to storage system. This approach allows source data to originate from both traditional input forms and from scannable forms as is most appropriate. It also affords improved error correction facilities for resolving unrecognized character problems with OCR processing.

5.3.3 Future Trends

It has been estimated* that at the end of 1974, the total U.S. installed base of data entry/communications facilities amounted to about 840,500 units, about half of which were buffered and unbuffered keypunching devices. However, the same source which predicts a 15 percent annual growth in keyboard based installations through 1979, predicts the following growth factors by 1979 as a multiple of 1974 installations:

. Multi-station keyboard to storage devices	-- Up 250 times
. Intelligent terminals	-- Up 7.5 times
. Soft-copy editing and conversational terminals	-- Up 3.4 times
. Hard-copy editing and conversational terminals	-- Up 1.2 times
. Stand-alone key to disk storage devices	-- Up 1.3 times
. Keypunches	-- Down 1.4 times
. Stand-alone key to tape devices	-- Down 7.7 times

There are presently some 3000 to 4000 OCR units and an unknown number of OMR units installed, OCR has not grown to the extent that was predicted in the 1960s when the first feasible system was developed. Unless procedures in the area of error recognition are vastly improved and/or more effective handwritten character readers are evolved, growth in the use of this approach will probably continue to be stunted.

OMR techniques are very effective in highly constrained and specialized situations. Modest growth in their use is probably to be expected.

As can be seen by the chart above, however, the most spectacular growth is expected to take place in the use of multi-station key/storage devices, with 250 keystations in use by 1979 for every one that was around in 1974.

Use of soft-copy CRT display terminals, presumably for real time application, will more than triple while hard-copy conversational terminals and stand-alone keyboard to disk and diskette units will show only modest gains through 1979.

As more installations switch to the newer techniques for key-entry the actual keypunch population will decline about 30 percent by 1979.

By far the biggest losers, however, are expected to be the stand-alone key to tape devices, with a decline of about 87 percent in installed keystations for the five-year period from 1974. As indicated earlier in this report, actual sales of key to compatible tape devices have been steadily declining since the early 1970s in favor of the easier to use alternatives offered by the key to cassette or cartridge or key to disk

*EDP Industry Report, published by International Data Corp., Waltham, Mass., 1975.

or diskette approaches in recent years will most certainly have a major impact on the key/cassette and key/cartridge market. After all, the diskettes are a thousand times faster (in search mode) than cassettes or cartridges, contain about the same amount of data and cost roughly the same.

These figures are also borne out by a spring 1975 Datapro survey regarding site intentions to change data entry facilities. Over half (52 percent) of the keypunch facilities and key to cassette facilities (58 percent) and two thirds of the key to compatible tape facilities (64 percent) plan to replace their devices in the immediate future, while only about 18 percent of the key to disk and key to diskette facilities contemplate any near-term changes. The actual intention figures, in percent, are shown below:

Users of the Following Types of Devices	Plan to Change to the Following Types of Devices				
	Online	Key/ Tape	Key/ Diskette	Multi-Station Key/Disk	Other
Keypunches	17	4	12	26	2
Key/Tape	30	-	16	18	2
Key/Cassette	0	0	42	8	8
Key/Diskette	11	0	--	4	4
Milti-Station Key/Disk	13	0	1	--	4

Another notable future trend is the further blending of many of the individually defined data-entry techniques for individual applications. By 1979 the categories of data entry devices as defined in this report will probably no longer hold. The advent of mixed-media systems and multi-station systems with remote data entry capabilities were breifly discussed in this report. To this must be added the evolving use of such facilities for remote batch entry, for accommodating sort and report generation, as well as other sophisticated software capabilities, and the ability of multi-station CPUs to communicate among themselves as well as with Touchtone telephones.

5.3.4 User Guidelines

While there are obviously no hard and fast rules of thumb in planning for data entry needs, in view of the technological discussions contained in the earlier sections of this report, and the future industry expectations as discussed in the preseding subsection, a number of comments can be made, most of which are self-evident:

- As with any computer system, total system costs for the data entry capabilities should be taken into account -- not just hardware device costs. Such costs above and beyond the hardware include:
  - personnel (operator and clerical support)
  - training
  - media handling and storage
  - office space utilization
  - data conversions
  - back-up procedures
  - computer preprocessing time
  - software programming
- Except for very small volume situations, unbuffered keypunches should normally be rejected in favor of other alternatives.
- Because of the obsolete media and technologies incorporated, key to punched tape and key to compatible magnetic tape entry devices should not normally be considered.
- Where key to diskette or key to disk pack facilities are cost justified, they should always be given greater consideration over key to cassette devices unless the application is limited to data capture and low volume editing.
- In general, all other things being equal, multi-station key to storage data entry devices or optical character recognition techniques are best suited for installations with high data entry volumes, while other stand-alone key to storage devices are more appropriate for lower volume installations.

For further guidance, the reader is referred to the matrix in Table 5.2.

5.4 Information System Alternatives

There are three basic hardware alternatives for fulfilling the general processing requirements for the software data repository. They are the utilization of:

1. The RADC Computer Center
2. Off-site facilities
3. A dedicated mini-computer

Below is a discussion of each of these alternatives highlighting the database management system availability. It is assumed that the alternatives have operational system software including an operating system, high order language compilers, and a set of utility programs.

Table 5. 2

Data Entry Options

KEY

XX	Not normally possible
X	Normally possible but poor application
O	Possible
●	Good use

	Keypunch/Key Verifier	Buffered Keypunches	Stand-Alone Key to Tape Devices	Stand-Alone Key to Cassette/Cartridge	Stand-Alone Key to Diskette Devices	Hard-Copy Teletypewriter Terminals	Soft-Copy Display Terminals	Multi-station Key to Disk Devices	Mixed Media Devices	Mark Sensing, OMR Devices	Optical Character Recognition Devices	Source Data Automation: POS, Data Coll. Term/ Tablets, etc.
Record Volume<200,000 records/month	O	O	O	O	O	O	O	X	X	O	X	O
Record Volume>200,000 records/month	X	X	O	O	O	X	X	●	O	O	●	O
Number of Characters/ typical record >80	XX	XX	O	O	O	O	X	O	O	O	O	O
Number of Format Levels/ typical job between 2 & 5	XX	O	X	O	O	X	X	O	O	O	O	O
Number of of Format Levels/ typical job >5	XX	X	X	X	O	X	X	O	O	O	O	O
Character Range: Numbers Only	O	O	O	O	O	O	O	O	O	●	O	O
Character Range: Standard Alphanumeric	O	O	O	O	O	O	O	O	O	X	O	O
Character Range: Extended Requirements - APL, lower case	XX	XX	O	O	O	O	O	O	O	XX	X	X
Possible to Capture Data at Source	X	X	X	O	O	O	O	X	O	O	O	●
Can Accomodate Forms & Procedures for OMR or OCR	X	X	X	X	X	X	X	X	O	●	●	O
Data Must be Entered with Minimal Delay	O	O	O	O	O	O	O	O	O	X	X	O

Table 5.2

Data Entry Options (cont'd)

KEY

XX	Not normally possible
X	Normally possible but poor application
O	Possible
●	Good use

	Keypunch/Key Verifier	Buffered Keypunches	Stand-Alone Key to Tape Devices	Stand-Alone Key to Cassette/Cartridge	Stand-Alone Key to Diskette Devices	Hard-Copy Teletypewriter Terminals	Soft-Copy Display Terminals	Multi-station Key to Disk Devices	Mixed Media Devices	Mark Sensing, OMR Devices	Optical Character Recognition Devices	Source Data Automation: POS, Data Coll. Term/, Tablets, etc.
Local Host Computer Supports Interactive Terminals	X	X	X	X	X	●	●	O	X	O	X	O
Hard Copy Required on Data Entry	O	O	O	O	O	O	X	O	O	O	O	O
Strong Editing Procedures Unavailable or Subsequent Detection of Errors by Central Computer Unacceptable	XX	XX	X	X	O	O	O	●	O	X	X	X
Rapid and Systematic Search of Recorded Data Records Required	XX	XX	O	O	●	O	O	●	O	X	X	O

5.4.1 The RADC Computer Center

The Information Sciences (IS) Division of RADC is currently maintaining two Honeywell 6180 computer systems to provide services for various computational needs at RADC. The GCOS and Multics Operating Systems are available.

Seven database management systems have been examined for possible use by the SDR. The system characteristics of each are summarized in Table 5.3 including the designation of the appropriate operating system, the organization responsible for maintaining the system at RADC, and an indication of whether its strengths are in the area of self-contained capabilities or host language capabilities. Below is a list of the major characteristics for each system. The current number of users is classified into four categories. None means that there are no current users, small implies less than 10 users, medium implies that there are between 10 and 50 users, and large means that the system is used by over 50 users.

For a definition of terms used below, see Appendix A.

DM-1 Characteristics [210,220]

. Data structure-Hierarchical (partially inverted)
. Host language
. Self-contained
. Number of users - None
. Special Features - Designed to provide data independence features.

JANUS Characteristics [218, 219, 223]

. Data structure - Relational
. No host language
. Self-contained
. Number of users - Small
. Special Features - Extensive statistical analysis package.

I-D-S Characteristics [214, 215]

. Data Structure - Network
. Host language
. Self-contained (limited)
. Number of users - Large
. Special Features - I-D-S/II consistent with the CODASYL/DDCL 73.[205]

MDQS Characteristics [212, 213]

. Data Structure - Network, Hierarchical, Singular
. No host language
. Self-contained
. Number of users - Medium
. Special Features - Commercial version of the World Wide Data Man-

Table 5.3

Database Management System Characteristics

Database Management System	Operating System	Maintenance	On-Site Maintenance	DBM Type
Data Manager - 1 (DM-1)	GCOS	--	--	Host Language
JANUS	Multics	MIT	--	Self-contained
Integrated Data Store (I-D-S)	GCOS	HIS	HIS	Host Language
Management Data Query System (MDQS)	GCOS	HIS	HIS	Self-contained
Force Management Information System (FMIS)	GCOS	SAC	--	Self-contained
Multics Relational Data Store (MRDS)*	Multics	HIS	--	Host Language
Multics Integrated Data Store (MIDS)*	Multics	HIS	--	Host Language

* These Systems are new Multics releases and their capability has not yet been established

agement System (WWDMS). Interfaces with I-D-S files.

FMIS Characteristics [226]

. Data structure - Fully inverted
. No host language
. Self-contained
. Number of users - Small
. Special features - Efficient retrieval capabilities.

MRDS Characteristics [217]

. Data structure - Hierarchical, Network, Relational
. Host language
. No self-contained
. Number of users - Small or none
. Special features - A function within the Multics Data Base Manager (MDBM).

MIDS Characteristics [217]

. Data structure - Hierarchical, Network
. Host language
. No self-contained
. Number of users - Small or none
. Special features - A function within the Multics Data Base Manager (MDBM).

5.4.2 Off-Site Facilities

Services can be purchased from commercial data centers, from installations connected to the ARPANET, and from commercial networks. This alternative offers a wide variety of possibilities through the many different hardware/software services available.

Attached to the ARPANET are a number of major large scale computers including IBM, CDC, DEC and Honeywell computers. Other network services offer a variety of processing capabilities. Some of the commercial network possibilities are:

- INFONET
    Computer Sciences Corporation
- DATANETWORK
    Honeywell Information System
- BCS NETWORK
    Boeing Computer Services
- UNA
    Utility Network of America
- TYMNET
    Tymshare, Inc.

The database management system alternatives are greatly expanded when considering to use one of the network services. Some of the better known DBMS are TOTAL, ADABAS, IMS, SYSTEM 2000 AND IDMS (Table 5.4) The basic capabilities are listed below.[230]

TOTAL Characteristics

. Data Structure - Network
. Host language
. No self-contained
. Number of users - Large

ADABAS Characteristics

. Data Structure - Network (partially inverted)
. Host language
. Self-contained
. Number of users - Large

IMS Characteristics

. Data structure - Hierarchical
. Host language
. Self-contained
. Number of users - Large

SYSTEM 2000 Characteristics

. Data structure - Hierarchical (partially inverted)
. Host language
. Self-contained
. Number of users - Large

IDMS Characteristics

. Data structure - Hierarchical, Network
. Host language
. No self-contained
. Number of users - Large

The cost to purchase these database management systems vary from $30,000 for a basic TOTAL System to $120,000 for ADABAS with full capability. However, if these systems have been purchased by the commercial network vendors themselves, the cost is distributed depending upon use.

The cost of service for both the hardware and software from these commercial networks varies with usage. The major cost elements are for on-line database storage, CPU and I/O usage, local data entry and remote batch terminals, and communication services. A gross estimate for a pilot facility based on previous projects offers a cost range of $8,000 to $19,000 per month.[114]

Table 5-4

Commercial Database Management Systems

Identification	Acronym	Computer Hardware System	Operating System	Originators	Initial Release Data
TOTAL	TOTAL	IBM 360 and 370 Honeywell Series 200 and 2000 Univac Series 70 and 9400/9700 NCR Century Series IBM System 3; CDC Cyber Series	DOS, DOS/VS, OS, VS1 and VS2 Mod 1 (MSR), Mod.2, and OS/2000 TDOS and DOS	Cincom Systems Inc.	Early 1969
Adaptable Data Base System	ADABAS	IBM 360, 370 or on an IBM-like Univac 9000 Siemens 4004	DOS, OS, OS/VS Disk Operating System (PBS for Siemens)	Software AG	1974 January
Information Management System	IMS/VS IMS-2	IBM 370 IBM 360, 370	OS/VS1, OS/VS2 OS/VS, OS/VS1, OS/VS2	International Business Machines North American Rockwell	1969 September
System 2000	System 2000	IBM360, 370 Univac 1106, 1108, 1110 CDC 6000 or Cyber 70	OS, OS/VS EXEC 8 SCORE, KRONOS	MRI Systems Corporation	1970 July
Integrated Database Management System	IDMS	IBM 360, 370 Univac Series 70	DOS, OS, OS/VS TDOS and DOS	Cullinane Corporation	1973 May

An estimate using Boeing Computer Services and the IDMS database management on a surcharge basis was developed. The assumptions made were that the application programs would occupy less than 100K bytes of primary storage, there would be 500 retrieval and update transactions per day, approximately 80 million characters of online storage would be required, and an estimated thirty (30) minutes per day of CPU time would be required for this level of activity. The total monthly cost estimate was $16,500 and was broken down as follows:

CPU and I/C Service	$ 6,000
IDMS Surcharge	2,400
One 3330 Spindle	1,300
Four dedicated ports	2,000
Low speed terminals (6)	550
Low speed terminal maintenance	150
RJE terminal and supplies	1,100
Communication service	3,000

5.4.3 A Dedicated Mini-Computer

The third alternative is the use of a dedicated mini-computer to perform the information processing for the SDR. Some mini-computer systems offer a subset of database management capabilities as discussed previously. These systems are termed file management systems and provide the capabilities for data entry and query, but have limited capabilities for database structuring and maintenance.

The following is a list of mini-computer systems that provide database management or file management capabilities.[114]

- Data General ECLIPSE
- General Automation 18/30
- Honeywell 60/62
- IBM System 3
- Hewlett Packard 3000
- Varian 75
- DEC PDP 11

A cost comparison for three mini-computer systems is provided in Table 5.5. The hardware/software configuration for each is listed below:

Varian 75/Vortex II/Total

. 128K words parity core memory
. Hardware floating point
. 16 line data communication multiplexor
. 93 million byte disk
. 75 ips, 800 bpi magnetic tape unit
. 800 lpm electrostatic printer/plotter
. Multiuser operating system
. TOTAL Database management system

Table 5.5

Mini-Computer Cost Comparison

Equipment	VARIAN		HP 3000		PDP 11/70	
	Purchase	Per Month	Purchase	Per Month	Purchase	Per Month
Hardware and Peripherals	$129,200	$ 3,590	$195,500	$ 5,430	$164,800	$ 4,580
Software and Firmware (Operating System & DBMS)	10,800	300	16,000	445	25,600	710
6 User Terminals (up to 9600 baud)	18,000	500	18,000	500	18,000	500
Site preparation	5,400	150	5,400	150	5,400	150
Maintenance	--	2,000	--	2,000	--	2,000
Supplies	--	300	--	300	--	300
Total Cost	$163,400	$ 6,690	$234,900	$ 8,825	$213,800	$ 8,240

HP 3000/MPE/IMAGE

- Model HP 32402C
- 128 K byte parity core memory
- Two 47 million byte disks
- 15 million byte system disk
- Scientific and Commercial Firmware
- 16 line multiplexor
- 500 lpm electrostatic printer plotter
- 45 ips, 800 bpi magnetic tape unit
- Multi-user operating system
- IMAGE/QUERY Database management system

PDP 11/70-IAS-IDMS

- Model 11/70 HA
- 128 K word parity core memory
- 512 K word swapping disk
- 88 million byte disk unit
- hardware floating point
- 45 ips, 1600 bpi tape unit
- 500 lpm electrostatic printer plotter
- 16 line multiplexor
- Multiuser operating system
- IDMS Database management system

5.5 Document Library Implementation Considerations

The previous sections assumed that the description of the documents would be stored in a computerized database (The Document Description Database). However, there are alternate methods for handling this information, their effectiveness dependent upon the size and utilization of the Library.

The primary need of the software document library is one common to all libraries - the need for an efficient storage and retrieval system.

In order to provide optimal satisfaction of users' research needs, any library requires:

1. A means of selecting documents containing information relevant to the purpose of its users.
2. A means of describing (indexing) such documents in an accurate consistent manner so that they can be subsequently retrieved upon the user's request.
3. Convenient storage and accessibility to both the documents and the indexes describing these documents.
4. The ability to manipulate indexed information for search and retrieval.

5. Some form of meaningful, organized output to describe the results of a search.

The Defense Documentation Center (DDC) provides a bibliographic service to eligible users through their Technical Report (TR) Program. The types of documents included in this program are the formally documented scientific and technical results of Defense-sponsored research, development, test and evaluation.

The Technical Report Program is a major source for documents related to Defense-sponsored projects in software engineering. To supplement these reports, other bibliographic services (e.g. ACM Computing Reviews) should be used.

The DDC reports are assigned an AD (Accessioned Document) number and are categorized into a two-level arrangement consisting of 22 major subject fields, with a further subdivision of 188 related subject groups. The subject groups related to software engineering can be further subdivided to reflect the special level of indexing needed for the repository.

Certain procedures are common to document processing within any storage and retrieval system regardless of whether it is a manual, semi-automatic or computer-oriented system. These procedures include the evaluation, description (indexing) and storage of documents. Procedures for the input, storage and manipulation of descriptive information, however, vary depending upon the particular system being employed. These procedures are described within the context of each system as it is discussed.

5.5.1 Manual Methods

Information listed on the Document Description Form (Figure 3.2) are transcribed to a series of 5x8 index cards. Several index files are established using this information. The primary files created are:

1. a complete bibliography - in the form of an abstract reference card. This card contains all the information found in the record and is filed numerically by accession number.
2. personal author and corporate author files - these cards lists author, title, source and date of publication and document accession number. These cards are filed alphabetically according to the author's surname or the corporate name.
3. a subject file - this file is set up in one of two ways:
   a. a keyword is listed as subject heading, followed by the author, title, source and date of publication, and the document accession number. If a document contains information covering more than one subject, a separate card is prepared for each keyword used. These cards are filed alphabetically by subject heading.

b. one card is prepared for each subject term in the index. On this card is listed only the accession numbers of all documents dealing with that particular subject. These cards are filed alphabetically by subject terms.

Manipulation and retrieval of indexed information becomes a difficult and inaccurate process as the size of the file and the number of search requests increase. A manual system becomes inadequate when documents must be searched with a combination of index terms. Each index term is searched individually, pulled from the file and then crossmatched against the other search terms. This type of manipulation increases the possibility for human error, especially in missing matches. The index list and bibliographic information is then typed in the form of a search report.

A manual system requires much typing and duplication of information. In addition to at least four different index cards for each document, all bibliographic and index lists would have to be typed by clerical personnel. These files are periodically updated to reflect current file status. Again, this requires considerable amount of retyping material already found in the file.

A manual system is recommended only when the document collection is comparatively small, and the search requests are few in number and not exceedingly specific.

5.5.2 Semi-Automated Methods

The semi-automated system differs from the manual system in its handling of the subject terms. An inverted-term matrix retrieval system is used for the subject terms.

An inverted-term matrix retrieval system involves the preparation of a file record for each concept of term to be indexed. The system used by the Reliability Analysis Center employs an 8-1/2 by 11 inch card (file record) for recording each concept. There are 10,000 possible items (documents) to which accession numbers are assigned. The preparation of one such concept card actually amounts to performing one complete search of the file. A request for the documents pertaining to a particular concept (i.e., cost estimation, reliability, modeling, etc.) is answered by listing the document accession number for which holes are drilled on the particular concept card. The great power of the method, however, lies in the ability to combine concepts and immediately read out the accession numbers pertaining to such combinations.

The advantages of this system are versatility, availability of convenient and inexpensive equipment, and savings in index scanning time resulting from the inverted nature of the index record.

The most important feature of the inverted-term-matrix indexing system is the versatility which it gives the indexer. This versatility arises from the fact that the indexer need not foresee all the possible combination of concepts which will be of interest to the inquirers. New concepts can be added to the index merely by adding new concept records. Old concepts can be easily modified or changed by eliminating from the file only those concept records which are involved. The system can even be used to find items which do not include a specific concept.

The disadvantages of this system are some of the same as those of the manual system. As there is still a need for, at a minimum, a bibliography file and a personal author file, there is the requirement of typing duplicate information.

5.5.3 Computer-Readable Methods

The information listed on the Document Description Form is keystocked and entered into the database. Although information storage and retrieval requirements do not require the full power of a database management system, the capabilities of defining updating and querying the database can be effectively utilized.

The output capabilities are greatly enhanced. In addition to the generation of a bibliography and reports on the results of literature searches, the following special reports can be produced:

- KWIC (Key Words in Context)

  In this report, significant words in each title are permitted and listed alphabetically so that each term serves as an access point to the particular document in which it appears.

- KWOC (Key Words Out of Context)

  This reports lists key words assigned to each document and cross-references each keyword to its document title. This index is quite useful in determining document content, since titles are often vague or do not always list all types which are covered in the document.

The advantages of a computer-readable system are:

- The ability to manipulate any number of index terms at one time, thereby providing depth and specificity which enhances the retrieval capabilities of the system

- The elimination of large amounts of time and repetitive clerical work involved in the generation of lists and creation of file cards

- The ability to provide the user with organized, meaningful reports which describe the results of a literature search

- The ability to generate a bibliography by automated means

- The ability to handle large volumes of information in a relatively short time

The disadvantages of a computer readable system are:

- The need for more highly qualified people to perform the task of data entry and simple retrievals

- Cost of utilizing the computer system

- Dependency on the availability of the computer system

5.5.4 Distribution Considerations

An important factor to consider when designing a document library is the level of document dissemination. The library can be a lending library, a distribution center, a reference library, or a combination of the above. Factors to consider are copyright permissions, proprietary and security needs, and the costs of reproduction and shipping.

6. Software Data Repository Design

6.1 Introduction

The functional model as presented in Figure 2.1 is the functional model of all the stages within the software data repository life cycle. As the repository evolves, it is expected that the information content of the data will change, the data collection, input processing, report production, consulting services, and the engineering and data analysis processes will expand, and that a greater quantity and a better quality of outputs will result.

A subset of the services and outputs described in Section 2 shall be provided through the implementation of a pilot facility. This pilot facility should be developed so that continous services be provided to the software development community as the pilot facility expands into a fully operational center.

This section presents the design for the Software Data Repository including the development and operation of a pilot facility and the expansion to a fully operational center.

6.2 Scope of the Pilot Facility

It is recommended that the Software Data Repository pilot facility be established as a separate contractor operated function and be administered and guided by the Rome Air Development Center (RADC), U.S. Air Force. The facility should be located in Rome, New York and utilize the computer systems available at RADC.

6.2.1 Source Data and Documents

The two major classes of information that form the inputs to the pilot facility shall consist of production/development data and textual information. The production/development data shall provide experience information on developing and maintaining computer software. The major sources of this data for the pilot facility include datasets that RADC has contracted to purchase and data that is acquired during the development and operation of the pilot facility.

The five known production/development dataset sources and their characteristics are listed in Table 6.1. The first three sources provide detailed testing and error data while the third source also provides some environment and resource utilization data. The last two sources provide summarized testing and error data in addition to environment, resource utilization, and production data.

The second class of information for the pilot facility shall consist of textual or expository information directly related to improving the process of designing, developing, testing and maintaining computer software. These documents include research and development reports, technical and journal articles, handbooks, conference proceedings, and specifications and standards.

Table 6.1

Production/Development Information Sources

	SOURCE				
Attributes	I	II	III	IV	V
Generic Name	Software Reliability Study	Data Acquisition	Modern Programming Practices	Management Data Collection and Reporting	Software Implementation Monitor
Abbreviation	SRS	DA	MPP	MDC+R	SIMON
Company/Agency/Program Contract#/job order number	TRW (Project III) F30602-74-C-0036	Raytheon (SAM-D) F30602-76-C-0140 Boeing (B-1) F30602-76-C-0140 Draper Labs (Apollo) F30602-76-C-0151 IBM (Safeguard) F30602-76-C-0161 IBM (PSL) F30602-76-0016	SAMTEC JON 5550 0847 Pave Paws JON 5550 0847 IBM (PSL) F30602-76-C-0016	WWMCCS sites	Mitre

The major sources for this information are:

A) Government reports relating to software engineering
   - Unclassified, unlimited distribution
     National Technical Information Service (NTIS)
   - Classified or limited distribution
     Defense Documentation Center (DDC)

NTIS is an agency of the U.S. Department of Commerce and is a central source for the public sale of Government-sponsored research, development and engineering reports and other analyses prepared by Federal agencies, their contractors or grantees.

DDC is the clearing house for the Defense Department's collection of research and development in virtually all fields of science and technology. Research and development activities within the U.S. Government and their associated contractors, subcontractors, and grantees, with current Government contracts, are eligible to receive most of the information from the DoD data banks located at DDC.

B) DoD software development specifications and standards
   - Naval Publications and Forms Center

All documents listed in the DoD Index of Specifications and Standards can be ordered from the Naval Publications and Forms Center. This DoD Index includes unclassified Federal, Military and Departmental specifications, standards, and related standardization documents, and those Industry documents which have been coordinated for DoD use.

C) All other documents related to software engineering
   - Author or Publisher

It is estimated that the number of textual documents for the pilot facility will be in the range of 2,500 to 3,500.

From the textual and production/development inputs, the Document Description Database, the Historical Database, and the Complementary Databases shall be generated.

6.2.2 Document Description Database

This database shall contain the information pertinent to describing the characteristics of the documents and the information necessary to retrieve the production/development data stored in the other two databases.

Approximately 1.5 million characters (without abstracts) are required for the database. If the abstracts were also included in the database, there would be an additional requirement of 3 million characters.

6.2.3 Historical Database

The Historical Database shall contain summarized information from the production/development reports. Approximately 1.4 million characters are required for the known production/development datasets (Table 6.2). This is expected to grow to approximately 3 million characters during the development and operation of the pilot facility.

6.2.4 Complementary Database

The Complementary Database shall contain detailed information from the production/development reports. Approximately 6.4 million characters are required for the known production/development datasets (Table 6.2). This is expected to grow to approximately 10 million characters during the development and operation of the pilot facility.

6.2.5 Output Types

The major pilot facility output types are briefly described in this subsection.

1. Historical and Complementary Database access
   Access to these two databases shall be available to RADC and facility personnel.

2. Data Subscription
   Computer readable data subsets shall be available for distribution to aid in research efforts that require productivity, cost, test, error, change, and software characteristics data. These shall be used to validate and refine software reliability and maintainability models and to aid in additional data analysis studies.

3. Data Compendiums
   Reports summarizing the database holdings shall be produced as facility publications.

4. A State-of-the-Art Report on Testing Tools and Techniques
   A report on various methods for testing computer software shall be produced including the identification, classification, description, and effectiveness of the testing tools and techniques.

5. A Software Engineering Glossary
   A comprehensive glossary of software engineering terms shall be produced.

6. Document Library
   Access to the Document Description Database and the source documents shall be available to RADC and facility personnel.

   Search services and reference information dissemination shall be extended to include other Government Agencies and approved contractors.

Table 6.2

Pilot Dataset Sizing

SOURCE

Entity	I Levels	I # Char	II Levels	II # Char	III Levels	III # Char	IV Levels	IV # Char	V Levels	V # Char	Total # Char
Historical Database											
Descriptions and Structure	PM	7,600	PM	37,500	PS	5,370	PSM	50,090	PSM	10,050	110,610
Environment	P	80			PS	5,150	PS	1,080	PS	180	6,490
Technology	M	7,400	P	750	PS	9,000	PS	3,000			20,150
Resource Utilization					PS	8,640	PSM	174,960	PSM	5,700	189,300
Production	PM	44,000	PM	165,660	PS	8,160	PSM	587,520	PS	4,400	809,740
Software Characteristics			PM	56,250	PS	3,600	PSM	178,200	PSM	25,200	263,250
TOTAL		59,080		260,160		39,920		994,850		45,530	1,399,540
Complementary Database											
Test Results	M	657,000	M	4,080,000	M	1,668,000		0		0	6,405,000
										TOTAL	7,804,540

Key

P	Project level
S	System level
M	Module level

The facility shall not act as a distribution center for the documents within its collection. Ordering information shall be disseminated to those requesting documents.

7. Software Engineering Bibliography
A bibliography shall be produced as a facility publication and include:

- annoted references to significant literature
- bibliographic citations for each reference
- indices for user searches including
  concept index
  subject term index (selected)
  corporate author index

6.2.6 Analysis of the Pilot Input Datasets

The five known source datasets for the pilot facility were analyzed to determined data content, and an analysis was performed to determine the data requirements for representative facility outputs. The analysis technique used was the generation of an input/output matrix listing group/attribute names for the SDR databases, an indication of the content of each source dataset, and an indication of the data requirements for the output products.

In Table 6.3 the first five columns reflect the data content for each source dataset, the last four columns denote the data requirements for utilization of the data. The sources are those listed in Table 6.1. The following is a discussion of each "use" column in the matrix and should be considered as examples of facility outputs.

A) Data Compendiums
Purpose - Provide summary information on the database holdings,
Input Requirements - Project and system type; person and computer type; standards, collection procedures and technology utilized; calendar, person, and computer time and cost; change attributes, quality and complexity measurements, and constituent attributes.

B) Reliability Data Subsets
Purpose - Provide data subsets for software reliability modeling efforts.
Input Requirements - System and module name and type; constituent attributes; testing, error, correction, and change information.

C) Complexity Data Subsets
Purpose - Provide data subsets for calculation of complexity measurements.
Input Requirements - System and module name and type; constituent attributes.

D) Cost Estimation Data Subsets
Purpose - Provide data subsets for cost estimation studies.
Input Requirements - Project and system type; person attributes;

BEST AVAILABLE COPY

Table 6.3

Input/Output Matrix

Group/Attribute Name	SOURCES[1] I	II	III	IV	V	USES[2] A	B	C	D
Project									
[1]Project ID	✓	✓	✓	✓	✓				
Name	✓	✓	✓	✓	✓				
Description			✓	✓					
Project Type						✓			✓
# Systems				✓	✓	✓			
# Modules	✓	✓	✓	✓	✓	✓			
System									
[1]System ID			✓	✓	✓		✓	✓	
Name				✓	✓			✓	
Description									
System Type						✓	✓	✓	✓
# Modules				✓	✓	✓		✓	
Module									
[1]Module ID	✓	✓	✓	✓	✓		✓		
Name (or ID)	✓	✓	✓	✓	✓				
Description									
Module Type						✓	✓		
Person									
ID #			✓	✓	✓	✓			✓
Phase			✓	✓	✓				
Person ID			✓	✓	✓				✓
Job function experience			✓	✓					✓
Application experience			✓	✓					✓
Rating criteria									✓
Rating measure									✓
Person level			✓	✓		✓			✓
Time percentage									
Travel requirements				✓					
Computer									
ID #	✓		✓	✓		✓			✓
Phase	✓		✓	✓					
Computer ID			✓			✓			
Hardware			✓	✓					
Software			✓						
Operating mode	✓		✓						
Response			✓	✓					✓
General									
ID #			✓			✓			✓
Phase			✓						
Contract type			✓						
Standards			✓			✓			
Collection purpose									
Collection procedures			✓	✓	✓	✓			
Constraints			✓						✓
Priorities									✓
Technology									
ID #	✓	✓	✓	✓		✓			✓
Phase	✓	✓	✓	✓					✓
Tech ID	✓	✓	✓	✓		✓			✓
Tool name	✓		✓	✓		✓			✓
Cost of usage									

[1] Table 6.1

[2] Section 6.2.4

BEST AVAILABLE COPY

Table 6.3

Input/Output Matrix (Cont'd)

	SOURCES[1] I	II	III	IV	V	USES[2] A	B	C	D
Calendar Time	✓		✓	✓	✓	✓			
ID #	✓		✓	✓	✓				
Phase				✓					
Estimated begin date			✓	✓					
Estimated end date				✓					
Actual begin date	✓			✓	✓	✓			
Actual end date	✓		✓	✓	✓	✓			
Person Time									
ID #			✓	✓	✓	✓			
Phase			✓	✓	✓				
Person level			✓	✓					
Estimated # person days			✓	✓	✓				
Actual # person days				✓	✓	✓			
Person Cost									
ID #				✓		✓			
Phase				✓					
Person level				✓					
Estimated person cost				✓					
Actual person cost				✓		✓			
Computer Time									
ID #			✓	✓	✓	✓			
Phase			✓	✓	✓				
Usage level			✓		✓	✓			
Estimated usage			✓	✓	✓				
Actual usage				✓	✓	✓			
Computer Cost									
ID #				✓	✓	✓			
Phase				✓	✓				
Estimated computer cost				✓	✓				
Actual computer cost				✓	✓	✓			
Travel									
ID #				✓					
Phase				✓					
Estimated # trips				✓					
Actual # trips				✓					
Travel Cost									
ID #				✓					
Phase				✓					
Estimated travel cost				✓					
Actual travel cost				✓					
Misc cost									
ID #				✓					
Phase				✓					
Estimated misc cost				✓					
Actual misc cost				✓					
Total Cost									
ID #			✓	✓	✓	✓			✓
Phase			✓	✓	✓				✓
Estimated total cost			✓	✓	✓	✓			✓
Actual total cost			✓	✓	✓	✓			✓
Documentation									
ID #			✓	✓					✓
Phase			✓	✓					✓
Document type			✓	✓					✓
# pages			✓	✓					✓
Reporting period			✓	✓					
Instructions									
ID				✓					✓
Phase				✓					✓
Instruction type				✓					✓
# instructions				✓					✓
Reporting period				✓					

[1] Table 6.1

[2] Section 6.2.6

BEST AVAILABLE COPY

Table 6.3

Input/Output Matrix (Cont'd)

	SOURCES[1]					USES[2]			
	I	II	III	IV	V	A	B	C	D
Changes									
ID #				✓		✓			
Phase				✓					
Change type				✓		✓			
# changes				✓		✓			
Reporting period				✓					
Test									
ID #		✓	✓	✓					
Phase		✓	✓	✓					
Test type		✓	✓	✓					
# tests		✓	✓	✓					
# failures		✓	✓	✓					
Reporting period				✓					
Error									
ID #			✓	✓	✓				
Phase			✓	✓	✓				
Error category type			✓	✓	✓				
Error category			✓	✓	✓				
# errors			✓	✓	✓				
Reporting period				✓					
Corrections									
ID #	✓	✓	✓						
Phase	✓	✓	✓						
Correction type	✓	✓	✓						
# corrections	✓	✓	✓						
Reporting period	✓	✓							
Quality									
ID #						✓	✓		✓
Phase							✓		✓
Quality measure type-major						✓	✓		✓
Quality measure type-minor						✓	✓		✓
Quality measure						✓	✓		✓
Method						✓	✓		
Complexity									
ID #			✓	✓	✓	✓		✓	✓
Phase			✓	✓	✓			✓	✓
Complexity measure type			✓	✓	✓	✓		✓	✓
Complexity measure			✓	✓	✓	✓		✓	✓
Testing									
ID #									
Phase									
Stress type									
Stress measure									
Constituents									
ID #	✓	✓	✓	✓	✓	✓	✓	✓	✓
Phase	✓	✓	✓	✓	✓		✓	✓	✓
Constituent type	✓	✓	✓	✓	✓	✓	✓	✓	✓
# of occurrences	✓	✓	✓	✓	✓	✓	✓	✓	✓
Test Info									
ID #	✓	✓	✓				✓		
Phase	✓	✓	✓				✓		
Test ID	✓		✓				✓		
Test type	✓		✓				✓		
Date run			✓				✓		
Time run			✓				✓		
Stress type		✓					✓		
Stress measure		✓					✓		
Test result		✓					✓		
Problem report number	✓	✓	✓				✓		
Report date	✓	✓					✓		

[1] Table 6.1

[2] Section 6.2.6

BEST AVAILABLE COPY

Table 6.3

Input/Output Matrix (Cont'd)

	SOURCES[1]					USES[2]			
	I	II	III	IV	V	A	B	C	D
Error Info									
ID #	✓	✓	✓				✓		
Phase	✓	✓	✓				✓		
Problem report number	✓	✓	✓				✓		
Error category type	✓	✓	✓				✓		
Error category	✓	✓	✓				✓		
Origin			✓				✓		
Severity type	✓		✓				✓		
Severity	✓		✓				✓		
Correction									
ID #	✓	✓	✓				✓		
Phase	✓	✓	✓				✓		
Problem report number	✓	✓	✓				✓		
Correction type	✓	✓					✓		
Date begun	✓						✓		
Date ended	✓	✓					✓		
Computer resources used		✓							
Person resources used									
#Instructions affected			✓						
#Pages documentation affected									
Change									
ID #							✓		
Phase							✓		
Change type							✓		
Change number							✓		
Date initiated							✓		
Date approved							✓		
Date completed							✓		
Person resources							✓		
Computer resources							✓		
Total cost							✓		
Change ID #							✓		

[1] Table 6.1

[2] Section 6.2.6

constraints, priorities, and technology attributes; total cost; amount of documentation and number of instructions produced; quality and complexity measurements; constituent attributes.

This list of data requirements is considered to be open ended because of the need to provide a variety of data compendiums and data subsets.

6.3 Pilot Information System Development and Recommendations

6.3.1 Introduction

This section presents the recommendations for the pilot information system and the development needed to establish an information system.

6.3.2 Developments for Input Processing

The input processing procedures discussed in Section 3 should be followed. However, the following developments must first occur.

6.3.2.1 Construction of a Glossary of Software Engineering Terms

So that there is a consistency of terminology to describe items in the database, there is a need for the construction of a glossary of software engineering terms. As part of the present study effort, a glossary of software engineering terms used in the literature was generated (Appendix C). This Glossary should be refined, and in conjunction with the work now being accomplished by the IEEE Subcommittee on Software Engineering Standards, be used as a guideline for indexing documents and standardizing the classification within the database definitions. In addition, this glossary will be published and disseminated as a facility publication.

6.3.2.2 Thesaurus Construction

Related to, but distinct from the construction of the glossary, is the need for the construction of a thesaurus for document indexing and retrieval.

The thesaurus should be developed using the following procedures;

1. Construct a set of thesaurus forms as illustrated in Figure 6.1 using the keywords (282) and the papers (59) contained in The Proceedings of the International Conference on Reliable Software.[401]

2. Generate a mini-thesaurus consisting of:
   - A structured term list grouping the terms into logical categories
   - An alphabetical term list which includes all the concept terms of the structured term list

THESAURUS FORM

Editor ____________________

Date ____________________

1. Keyword: ____________________

2. Synonyms: ____________________

____

____

____

3. Related Terms: ____________________

____

____

____

4. Broader Terms: ____________________

____

____

5. Narrower Terms: ____________________

____

____

6. Definition: ____________________

____

____

Figure 6.1 Thesaurus Form

3. Construct an additional set of thesaurus forms using this mini-thesaurus and the documents in the present library.[402]

4. Update the mini-thesaurus to generate a thesaurus that will serve as a basis for subsequent document indexing and retrieval.

6.3.2.3 Input Forms and Data Formats

Standard input forms and procedures must be designed to be compatible with the databases. The attributes of the database must be defined in the data definition language of the database management system.

6.3.2.4 Analysis of Input Datasets

Procedures must be established to evaluate the scope and content of the input datasets. The purposes of this evaluation are to determine the applicability of the data, to determine the requirements for further processing to transform the data to the SDR formats, and to determine the sensitivity of the data.

6.3.2.5 Data Translation

All of the known source datasets for the pilot facility have some or all of the data in computer readable form. It is expected that additional datasets will also be in computer readable form.

Procedures must be established for transforming these datasets into the SDR database formats. It is recommended that general purpose translation software be developed along the lines of the data translation tools developed by the University of Michigan Data Translation Project.[202] These tools provide for describing both the source and target data, and the necessary transformations required to generate the target data from the source.

For those datasets that arrive in hard copy form and contain small amounts of data (such as the system productivity and cost data from the MPP Projects), it is recommended that manual translation be performed to be compatable with the SDR databases.

In addition to conversion of these foreign files into the data format for the pilot, code conversion and data summarization procedures are required.

6.3.2.6 Security Procedures

The facilities of the operating system and the database management system shall be used to provide backup and restart and recovery capabilities to protect the database and the application programs from destruction or loss. It is recommended that no corporate sensitive data be accepted for the pilot facility.

6.3.3 System Recommendations

From the system alternatives presented in Section 5, the following recommendations are made.

6.3.3.1 Data Collection Systems

As there is no one data collection system that collects a complete set of data, it is recommended that data from various data collection systems be acquired and evaluated. It is also recommended that during the development of the pilot facility, the specifications for a data collection system be generated.

6.3.3.2 Data Entry

As the majority of the known source datasets for the pilot will be in computer readable form, the requirement for keystroking data is minimal.

Therefore, it is recommended that existing keypunches be used to convert this data to computer readable form.

6.3.3.3 Information System

It is recommended that the RADC Honeywell 6180 Computer System using the GCOS operating system and the MDQS database management system be used for the implementation and retrieval of the facility databases. This alternative provides easy to use production oriented hardware and software services to both RADC and facility personnel to assist in their research efforts.

It is recommended that during the operation of the pilot facility the ARPA Network not be used to provide database retrieval and updating capabilities. However, it is recommended that a task be undertaken to determine the feasibility of using the ARPA Network as the facility expands to a fully operational center.

6.3.3.4 Document Library

Because of the expected number of documents in the document library and the need for a versatile information storage and retrieval capability, it is recommended that the description of the documents be stored in a computer-readable Document Description Database.

6.3.4 Data/Document Acquisition

Some datasets and textual documents have been identified to form the basis for the pilot facility. However, to insure continued acquisition of data for the pilot and beyond, there is a need for the establishment of a data acquisition program. The methods and procedures for the identification and acquisition of the document inputs must be established during the development of the pilot facility to assure quality data acquisition from all segments of the government-industry software development community.

Procedures for the acquisition of the textual information can be accomplished routinely with minimal engineering effort. The acquisition of production/development data requires more detailed procedures that assist in identifying potential sources, solicitation of these sources, and the acquisition of the data from these sources. Because of economic and proprietary considerations, this acquisition is more difficult.

6.4 Pilot Facility Operation

It is recommended that the responsibility for the operation of the pilot facility be divided into four major functional areas (Figure 6.2).

1. Technical and administrative direction
2. Document acquisition
3. Information processing
4. Data and information analysis

6.4.1 Technical and Administrative Direction

The technical and administrative direction involves the overall responsibility for all contractor operating and policy decisions, and provides official interface to the RADC Project Engineer on technical matters.

The three remaining functional areas are technically oriented with each reporting to a project manager. These functional areas are accountable to each other.

6.4.2 Data/Document Acquisition

The function of data/document acquisition during the operation of the pilot facility encompasses the identification, solicitation, and acquisition of data and documents and the refinement of the methods and procedures for this acquisition process. There is a functional separation between the acquisition of textual information and production/development reports.

6.4.3 Information Processing

There are three major, but interrelated, functional areas included for information processing.

1. The input processing function includes document review and indexing, document library and database updating, and transformation and summarization of the production/development reports.

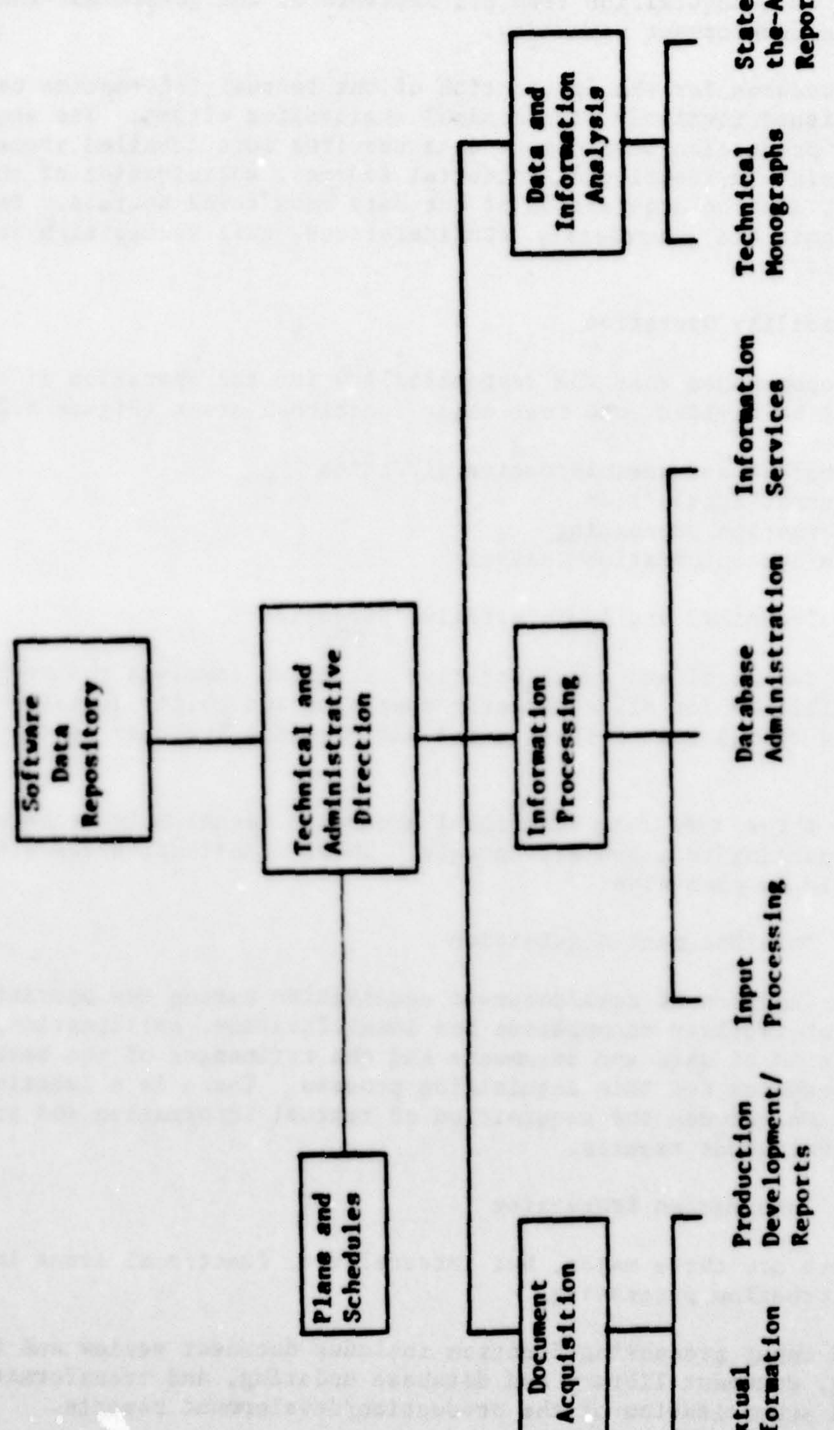


Figure 6.2 Pilot Operation Organization

2. The database administrative function includes controlling the contents and integrity of the databases by establishing and modifying database maintenance procedures, evaluating the utilization and growth characteristics of the database, restructuring the databases, monitoring and evaluating the security procedures, and by establishing user interfaces to the database and the requirements for data transformations.

3. The information services function involves the evaluation of users needs including dissemination of information on services provided by the facility; the production of data compendiums, subsets of the database, and a bibliography; assisting in search and retrieval requests; providing answers to "simple" user inqueries and generating requirements for in-depth analysis requests.

6.4.4 Data and Information Analysis

This function involves performing data and engineering analysis and producing technical monographs and state-of-the-art reports containing the results of the research efforts.

At a minimum, for the pilot facility, this includes producing a report that summarizes and evaluates the capabilities of tools and techniques that have been developed for testing software.

Other efforts that should be considered include software reliability and maintainability model development, and data analysis studies on software complexity, quality, productivity, and cost estimation.

6.5 Pilot Facility Management Plan

The recommended schedule for the development and operation of the pilot facility for a two-year period is illustrated in Figure 6.3.

The facility staff should consist of both professional and non-professional personnel. It is recommended that the professional staff consist of individuals from the following personnel categories:

- Manager
- Senior Analyst
- Software Engineer
- Information Scientist

and that the non-professional staff consist of clerical personnel and technicians.

The capabilities of the manager are required for overall administration and client communication.

The principal function of the software analyst(s) is database administration. This includes the establishment of the database formats and the

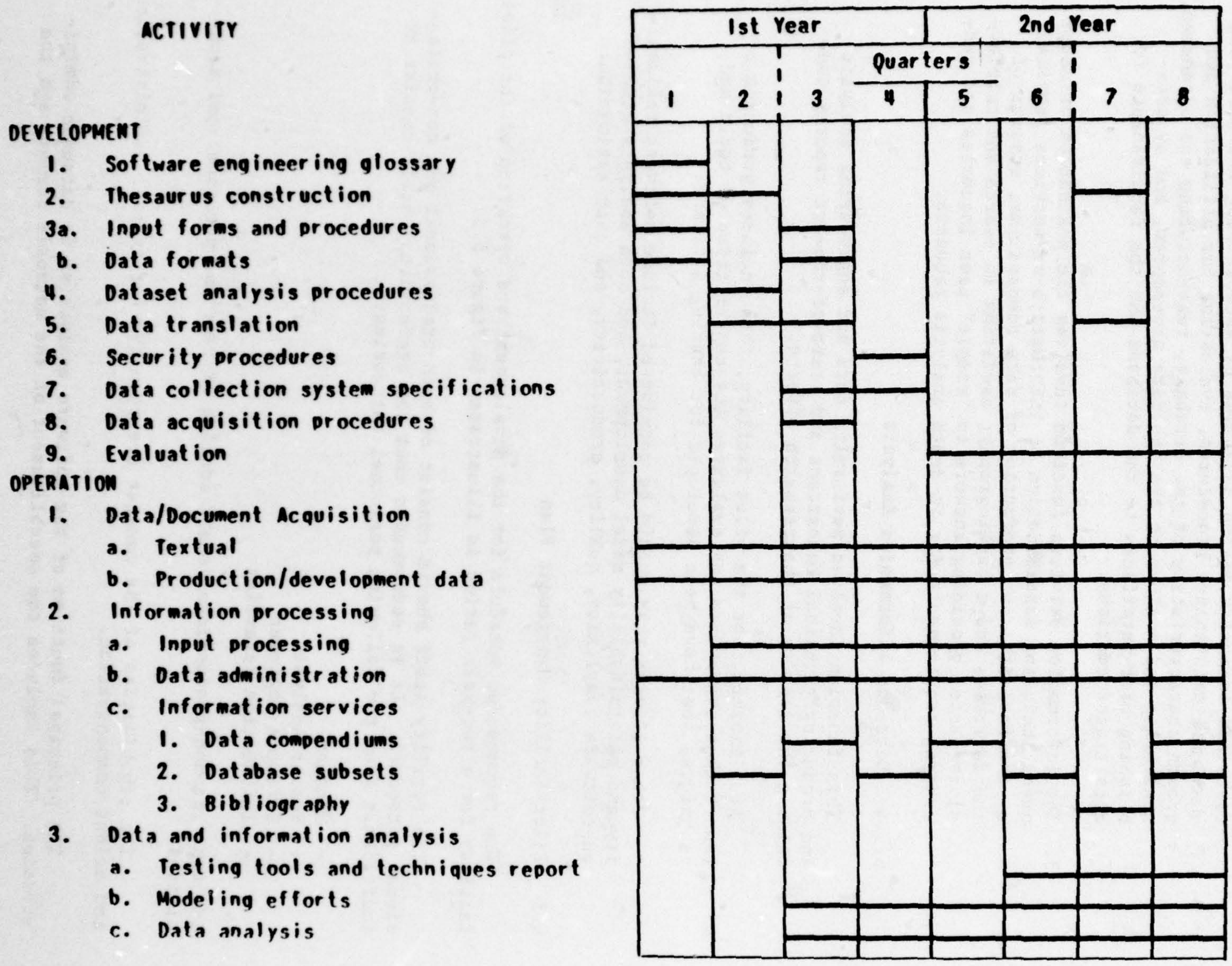


Figure 6.3 Pilot Facility Schedule

dataset analysis procedures, the establishment and evaluation of the data translation and security methods, and the maintenance and overall responsibility for the databases.

The software engineering function includes the production of state-of-the-art reports, technical monographs, and the data collection system specifications. It also includes the establishment of the data acquisition procedures for the production/development data, and the acquisition of the data. The software engineers shall provide consulting services for the information scientist and the senior analyst, develop computer programs to supplement the database management system, and shall have the overall responsibility for the production of the data compendiums and the database subsets.

The information science functions include the construction of a glossary and a thesaurus, the establishment of the input forms and procedures, the establishment of the document acquisition procedures for the textual information, the indexing of the documents, the production of the Bibliography, providing search and reference services, the acquisition of the textual documents, and the overall responsibility for the document library.

The clerical personnel are responsible for daily typing and report typing, some input processing functions, order processing and shipping, telephone services, travel arrangements, etc.

The technicians are responsible for computer data entry, data summarization, and ordering documents.

6.6 Pilot Facility Evaluation

The evaluation of the pilot facility includes the determination of the effectiveness of the facility in terms of responsiveness to the user's needs and the operational flexibility and efficiency of the facility.

The evaluation of the internal operation of the facility must consider the amount of time to process the source input and the amount of time to answer user inqueries. The amount and quality of the data and information dissemination must also be a consideration.

The users of the pilot facility shall be RADC personnel, facility personnel, and approved RADC contractor and other Air Force personnel. The responsiveness to these users' needs can be determined by both quantitative and qualitative measures. The quantative measures include the number of requests for data subsets and reports, the number of inqueries and bibliographic services, the percentage of these requests that could be fulfilled, and some measure of the usefulness of the services and products in terms of resources saved. This last figure is subjective in that the users themselves must determine the relative savings from using the products and services.

The qualitative measures include feedback from the users and potential users concerning the quality of the products and services, and what additional services they require.

This kind of feedback is useful in determining what additional services should be provided and/or other areas that could be included in a fully operational center. This feedback can be acquired through participation in government and industry software engineering symposiums, conferences, meetings, and committee activities.

It is recommended that the development, operation, and evaluation of the pilot facility proceed in such a manner that a fully operational center can evolve to provide continuous and expanded services to the users. As new services are provided and internal processing functions are expanded, they should be accomplished with minimum disruption on the operation of the repository, whether it be the pilot facility or a fully operational center.

6.7 A Fully Operational Center

As the pilot facility evolves into a fully operational center, the user community shall expand, the scope of the information shall change, and the engineering analysis activities should encompass more software engineering areas which will result in a better quality and greater quantity of output services.

6.7.1 The User Community

It is expected that as the repository evolves, the user community will encompass more members of the software development community including not only government agencies and contractors, but the private sector as well.

This expansion of the user community necessitates the need for the incorporation of a cost recovery program. This program can consist of charging the user an annual cost which would entitle the user to publications issued during the period plus a prescribed number of direct inqueries. In addition, separate pricing for individual publications and consulting services can be instituted.

To provide existing and potential users with an awareness of the services provided, the following vehicles can be used:

1. Brochures listing repository publications and services along with complete pricing and ordering information.
2. Newsletters discussing repository services and other software engineering activities.
3. Workshops concentrating on specialized software development problems, trends, etc.
4. Presentations at software engineering conferences, technical meetings, etc.

6.7.2 Information Content

The content of the databases and the number of documents in the library will increase as the repository expands.

AD-A050 636   IIT RESEARCH INST CHICAGO ILL   F/G 9/2
SOFTWARE DATA REPOSITORY STUDY.(U)
DEC 76 L M DUVALL   F30602-75-C-0257
UNCLASSIFIED   IITRI-E6330-28   RADC-TR-76-387   NL

2 OF 2
AD A050636

END
DATE
FILMED
4-78
DDC

The primary information content of the databases for the pilot facility consists of testing, error, and change data. It is expected that as software reliability and maintainability models are developed and validated, the need for the data will decrease, but will still exist.

With an aggressive data collection and acquisition program, more cost and productivity data should become available.

The initial estimate for the size of the pilot facility databases is approximately 16 million characters. By being selective in the acceptance of data, purging obsolete data, and including productivity and cost data in the Historical Database at only the project and system levels, the size of the databases should not exceed 100 million characters.

Because of this change in information content, one or more of the following activities should occur:

1. Database reorganization
2. Addition of new data items
3. Purging obsolete complementary database entities and constructing new complementary database entities.
4. Data summarization modification.
5. Modifying input processing, dataset analysis, and data translation procedures.
6. Establishing a secure information system.

6.7.3 Output Services

The output services should be expanded to include the results of more in-depth studies than were produced during the pilot facility. This can be accomplished because of the expanded scope of the data, and because we will have learned more about the software development process and the techniques to analyze the data.

Database and document library access shall continue to be available to all qualified users. Data subsets and data compendiums shall be produced as user requirements are identified while a handbook on designing and developing computer software shall be developed to assist both the practitioner and the researcher.

Technical monographs and state-of-the-art reports shall be produced as user requirements are identified. It is expected that they will cover the following areas:

- cost and resource estimation
- software development monitoring
- quality software production
- software quality measures
- software complexity measures
- software requirements and specifications

The inquery and consulting services shall become an increasingly more important function as the user community expands, more data becomes available, and as greater expertise exists in the repository.

These services may take the form of answering questions on the use of a data compendium to long term consulting actions. To provide quality consulting services, it shall be necessary for the software engineers to specialize in certain areas. These areas may be reliability, maintainability, testing tools and techniques, development techniques, requirement and specification methodologies, and/or project monitoring.

The consulting tasks may include measurement, estimation, and prediction of the reliability of a software system; assist in determining requirements and test plans; conduct broad and indepth surveys among government and industrial organizations to seek out specialized information and data; and perform qualitative and quantitative studies on estimating the resources for a software development project.

7. Recommendations

The following recommendations are a summary of the recommendations contained in this report.

- Establish a software data repository to provide users with access to quality, relevant, software development experience information to assist in their research and development efforts.

- Develop the repository so that continuous service be provided to the users.

- Include both production/development data and textual information. Establish three databases including:

  - o Document Description Database containing descriptions of the documents and the data.
  - o Historical Database containing summarized productivity, cost, environment, test, error, and change data.
  - o Complementary Database containing detailed test, error, and change data with the capability to add additional entities.

- Provide the following services and outputs:

  - o Database and Document Library access
  - o Database Subsets and Data Compendiums
  - o Technical Monographs and State-of-the-Art Reports
  - o Handbooks and a Bibliography
  - o Inquery and Consulting Services
  - o Data and Engineering Analysis

- Establish a pilot facility as a subset of a fully operational center. The following are the recommendations for the establishment and operation of a pilot facility:

  - o The facility be located at RADC in Rome, NY
  - o Limit the users to members of the Air Force and approved contractors and agencies.
  - o Accept only non-sensitive data including RADC purchased datasets.
  - o Construct a glossary of software engineering terms and a thesaurus.
  - o Establish procedures for acquiring, evaluating, summarizing, and transforming the production/development data.
  - o Acquire data from various data collection systems. Establish the specifications for a data collection system for the repository.
  - o Implement the databases on the RADC Honeywell 6180 computer system using the GCOS operating system and the MDQS database management system.
  - o Do not use the ARPA Network for the pilot facility. Determine feasibility for using the ARPA Network for a fully operational center.

- Produce data compendiums and data subsets containing productivity, error and cost data; a bibliography of relevant software engineering literature; technical monographs containing the results of modeling and data analysis efforts.
- Evaluate the facility and determine additional user requirements.

- Evolve into a fully operational center:

  - Expand the user community.
  - Extend scope and content of the data.
  - Expand output services.

8. Bibliography

The Bibliography is divided into five series to facilitate easy access to different subject areas.

The major subject areas and their corresponding reference numbers are illustrated below.

001-099	Software Engineering
101-116	Data Center Operations
201-230	Database Management and Data Translation
301-309	Computer Networks and Security
401-404	Bibliographies

SOFTWARE ENGINEERING

Series 001

001 Amory, W., Clapp, J.Q., "Engineering of Quality Software Systems (A Software Error Classification Methodology)", RADC-TR-74-325, Volume VII January 1975, (A007772).

002 Baker, F.T., "Structured Programming in a Production Programming Environment", IEEE Transactions on Software Engineering, June 1975, Volume SE-1, No. 2

003 Barry, Barbara S., "Structured Programming Series, Volume X, Chief Programmer Team Operations Description", 22 January 1975, AD-A008 861, RADC-TR-74-300

004 Bell, D.E., Sullivan, J.E., "Further Investigations into the Complexity of Software", MTR-2874, Volume II, 30 June 1974

005 Boehm, B.W., Bosch, C.A., Liddle, A.S., Wolverton, R.W., "The Impact of New Technologies on Software Configuration Management", TRW Systems Group, 10 June 1974

006 Boehm, B.W., Brown, J.R., Kasper, H., Lipow, M., MacLeod, G.J., Merritt, M.J., "Characteristics of Software Quality", TRW-SS-73-09, 28 December 1973

007 Boehm, Barry W., "Software and Its Impact: A Quantitative Assessment", Datamation, May 1973, pp. 48-59

008 Boehm, Barry W., "Software Design and Structuring", Practical Strategies for Developing Large Software Systems, Ellis Horowitz Ed., Addison-Wesley 1975

009 Boehm, Barry W., "Some Steps Toward Formal and Automated Aids to Software Requirements Analysis and Design", Proceedings of IFIPS 74, August 1974, pp. 192-197

010 Bolsky, M.I. and Shooman, M.L., "Types, Distribution and Test and Correction Times for Programming Errors", Proceedings International Conference on Reliable Software, 21-23 April 1975, IEEE Catalog No. 75CH0940-7CSR

011 Bratman, H., Court, T.D., and Raden, B.R., "The SDC Software Factory User's Guide, Impact", System Development Corporation publication, 15 July 1975

012 Bratman, H., Cudney, P.F., Johnson, B.G., and Shinn, L.A., "The SDC Software Factory", System Development Corporation publication, 1 August 1973

013 Bratman, Harvey and Court, Terry, "The Software Factory", Computer Magazine, pp. 28-37, May 1975

014 Brooks, F.P., "The Mythical Man-Month", Datamation, December 1974, pp. 45-52

015 Brooks, F.P., "The Mythical Man-Month, Essays on Software Engineering", Addison-Wesley 1975

016 Brooks, N.B. and Gannon, C., JAVS Computer Program Documentation: User's Guide, August 1975

017 Brooks, N.B. and Gannon, C., JAVS Computer Program Documentation: Reference Manual, August 1975

018 Carter, D.M., Gibson, H.L., Rademacher, R.A., "A Study of Critical Factors in Management Information Systems for the U.S. Air Force", Colorado State University, Information Systems Series 460-2

019 Clapp, J.A., "Software Engineering: Problems and Future Developments", November 1974, MTR-2791, ESD-TR-74-195

020 Clapp, J.A., "Monitoring Software Development for Reliability Indications", IEEE Electronics and Aerospace Systems Convention, pp. 38-42, October 1974

021 Clapp, J.A. and Sullivan, J.E., "Automated Monitoring of Software Quality", AFIPS Conference Proceedings Vol 43, 1974 National Computer Conference, pp. 337-341

022 Corrigan, A.E., et. al., "Specifications for Simon: A Software Implementation Monitor", MTR-3056, July 1975

023 Corrigan, Ann E., "Results of an Experiment in the Application of Software Quality Principles", MTR-2874, Vol. III, 30 June 1974

024 Coutinho, John de S., "Software Reliability Growth", Record 1973 IEEE Symposium on Computer Software Reliability, April 1973, pp. 58-64

025 Craig, G.R., Hetrick, W.L., Liplow, M., Thayer, T.A., et. al., "Software Reliability Study", RADC-TR-74-250, October 1974, AD 787 784

026 Culpepper, L.M., "A System for Reliable Engineering Software", Naval Ship Research and Development Center, November 1974, AD A-003 465

027 DeRoze, Barry C., "An Introspective Analysis of DoD Weapon System Software Management", Defense Management Journal, Vol. 11, No. 4, October 1975, pp. 2-7

028 Dickson, J.C., Hesse, J.L., Kientz, A.C. and Shooman, M.L., "Quantitative Analysis of Software Reliability", Proceedings 1972 Annual Reliability and Maintainability Symposium, January 1972, pp. 148-157

029 Fleischer, R.J., "Engineering of Quality Software Systems (Effects of Management Philosophy on Software Production)", RADC-TR-74-325, Volume II, January 1975, (A007767).

030 Fleischer, R.J., "Software - First System Design", Spring COMPCON 76 - Digest of Papers, February 1976

031 Fleischer, R.J., Spitler, R.W., "A Project Management System for Software Development", Mitre Corporation Abstract, pp. 1-23

032 Flynn, Robert J., "Design of Computer Software", Proceedings 1975 Annual Reliability and Maintainability Symposium, January 1975, pp. 476-479

033 Goodenough, J.B. and Gerhart, S.L., "Toward a Theory of Test Data Selection", IEEE Transactions on Software Engineering, June 1975, Volume SE-1, No. 2

034 Goodenough, John B. and Ross, Douglas T., "The Effect of Software Structure on Software Reliability , Modifiability, Reusability and Efficiency: A Preliminary Analysis", Softech Inc., December 1973, AD 780 841

035 Hall, Robert R., "MIPS Measurement Forms for ASTROS", ASTR-1287-3591-00, June 1975

036 Hecht, Herbert, "Can Software Benefit from Hardware Experience?" Proceedings 1975 Annual Reliability and Maintainability Symposium, January 1975, pp. 480-484, AD 004 076

037 Johnson, J.P., "Software Reliability Measurement Study", SAMSO-TR-75-279, 8 December 1975

038 Keirstead, R.E., "On Software Certification", Spring COMPCON 76 - Digest of Papers, February 1976

039 Kerninghan, B. and Plauger, P.J., "Software Tools", Proceedings of the 1st National Conference on Software Engineering, September 1975, IEEE Catalog No. 75CH0992-8C

040 Kessler, Marvin M., Kister, William E., "Structured Programming Series Volume XIV, Software Tool Impact", 22 May 1975, RADC-TR-74-300, (015795).

041 La Padula, L.J., "Engineering of Quality Software Systems (Software Reliability Modeling and Measurement Techniques)", RADC-TR-74-325, Volume VIII, January 1975, (016415).

042 Littlewood, B. and Verrall, J.L., "A Bayesian Reliability Growth Model for Computer Software", Record 1973 IEEE Symposium on Computer Software Reliability, April 1973

043 Luppino, F. M. et. al., "Structured Programming Series, Volume V, Programming Support Library (PSL) Functional Requirements", 25 July 1974, AD/A 003 339, RADC-TR-74-300

044 Manley, John H., "Embedded Computer System Software Reliability", Defense Management Journal, Vol. 11, No. 4, October 1975, pp. 13-17

045 McCarthy, Rita, "Applying the Technique of Configuration Management to Software", Defense Management Journal, Vol. 11, No. 4, October 1975, pp. 23-28

046 McClure, C.L., "Top-Down, Bottom-Up and Structured Programming", Proceedings of the 1st National Conference on Software Engineering, September 1975, IEEE Catalog No. 75CH0992-8C

047 McCoy, Bruce J., "New Flight Software Development Techniques -- The DAIS Example", NAECON '75 Record, pp. 234-243

048 McHenry, Robert C., "Software Development Process Revisions", How to Make Computers Easier to Use, Digest of Papers, IEEE Catalog No. 75CH0988-6C, September 9-11, 1975

049 Meeker, Ralph Edward Jr., and Ramamoorthy, C.V., "A Study in Software Reliability and Evaluation", Technical Memorandum No. 39, pp. 58-64, 15 February 1973

050 Miller, Edward F., Jr., "Overview and Status", General Research Corporation publication, 1 October 1974

051 Miller, Edward F., Jr., "RXVP: FORTRAN Automated Verification System, Level 1, System Summary", General Research Corporation publication, 1 October 1974

052 Miller, Edward F., Jr., "RXVP: FORTRAN Automated Verification System, Level 1, User's Guide", General Research Corporation publication, 1 October 1974

053 Mills, H.D., "How to Write Correct Programs and Know It", Proceedings International Conference on Reliable Software, 21-23 April 1975 IEEE Catalog No. 75CH0940-7CSR

054 Mills, Harlan D., "On the Development of Large Reliable Programs", 1973 IEEE Symposium on Computer Software Reliability, May 1973, pp. 155-160

055 Moranda, P.B., "Prediction of Software Reliability During Debugging", Proceedings 1975 Annual Reliability and Maintainability Symposium, January 1975, pp. 327-332

056 Naughton, John J., "Structured Programming Series, Volume XIII, Final Report", 15 July 1975, RADC-TR-74-300, (A020858).

057 Ortega, Louis H., "Structured Programming Series. Volume VII, Documentation Standards", 21 September 1974, AD A008 639, RADC-TR-74-300

058 Parker, Richard O., "RXVP: An Automated Verification System for FORTRAN", General Research Corporation publication, February 1975

059 Parnas, D.L., "The Influence of Software Structure on Reliability", Proceedings International Conference on Reliable Software, 21-23 April 1975, IEEE Catalog No. 75CH0940-7CSR

060 Ramamoorthy, C.V., Cheung, R.C. and Kim, K.H., "Reliability and Integrity of Large Computer Programs", March 1974, AD 779 339

061 Ramamoorthy, C.V. and Ho., S.-B., F., "Testing Large Software with Automated Software Evaluation Systems", IEEE Transactions on Software Engineering, Volume SE-1, No. 1, March 1975

062 Ramamoorthy, C.V. and Kim, K.H., "Software Monitors Aiding Systematic Testing and their Optional Placement", Proceedings of the 1st National Conference on Software Engineering, September 1975, IEEE Catalog No. 75CH0992-8C

063 Ramamoorthy, C.V., Meeker, R.E., Jr., and Turner, J., "Design and Construction of an Automated Software Evaluation System", Record 1973 IEEE Symposium on Computer Software Reliability, April 1973, pp. 28-37

064 Reifer, D.J., "A Structured Approach to Modeling Computer Systems", 30 August 1974, AD/A 004 075

065 Reifer, D.J., "Automated Aids for Reliable Software", Report SAMSO-TR-75 183, 26 August 1975

066 Reifer, D.J., "Automated Aids for Reliable Software", Proceedings International Conference on Reliable Software, 21-23 April 1975, IEEE Catalog No. 75CH0940-7CSR

067 Reifer, Donald J. and Ettenger, R. Lee, Lt., "Test Tools: Are They a Cure-All?" Proceedings 1975 Annual Reliability and Maintainability Symposium, January 1975, pp. 492-497, AD/A 004 078

068 Richards, R. Russell, "Computer Software: Testing, Reliability Models, and Quality Assurance", Naval Postgraduage School, July 1974, AD/A 001 260

069 Ronback, James A., "Software Reliability How it Affects System Reliability", 1975 Canadian SRE Reliability Symposium, 10 May 1975

070 Ross, Douglas T., Goodenough, John B. and Irvine, C.A., "Software Engineering: Process, Principles and Goals," Computer, May 1975

071 Rubey, R.J., "Quantitative Aspects of Software Validation", IEEE Transactions on Software Engineering, June 1975, Volume SE-1, No. 2

072 Schick, G.J. and Wolverton, R.W., "Achieving Reliability in Large Scale Software Systems", Proceedings 1974 Annual Reliability and Maintainability Symposium, January 1974, pp. 302-319

073 Schneidewind, N.F., "Analysis of Error Processes in Computer Software", Proceedings International Conference on Reliable Software, 21-23 April 1975, IEEE Catalog No. 75CH0940-7CSR

074 Schneidewind, Norman F., "An Approach to Software Reliability Prediction and Quality Control", AFIPS Conference Proceedings Vol. 41, Fall 1972, pp. 837-847

075 Schwartz, Jules, I., "Construction of Software: Problems and Practicalities", Practical Strategies for Developing Large Software Systems, Ellis Horowitz, Ed., Addison Wesley 1975

076 Scott, R.F. and Simmons, D.B., "Predicting Programming Group Productivity - A Communication Model", Proceedings of the 1st National Conference on Software Engineering, September 1975, IEEE Catalog No. 75CH0992-8C

077 Shura-Bura, Mikhail R., "Software Implementation Studies: Problems and Prospects", Proceedings of IFIPS 74, August 1974, pp. 427-430

078 Slaughter, John B., "Understanding the Software Problem", AFIPS Conference Proceedings, Vol. 43, 1974 National Computer Conference, pp. 333-336

079 Smith, Ronald J., "Structured Programming Series, Volume IX, Management Data Collection and Reporting", 24 October 1974, AD A008 640, RADC-TR-74-300.

080 Smith, Ronald L., "Structured Programming Series, Volume XI , Estimating Software Project Resource Requirements," 22 January 1975, RADC-TR-74-300 , (A016416).

081 Stillman, Rona B. and Leong-Hong, Belkis, "Software Testing for Network Services", NBS Technical Note 874, July 1975

082 Stucki, L.G., "Automatic Generation of Self-Metric Software", 1973 IEEE Symposium on Computer Software Reliability, May 1973, pp. 94-100

083 Stucki, Leon G., "Automated Tools and Techniques Assisting in Software Development", Practical Strategies for Developing Large Software Systems, Ellis Horowitz, Ed., Addison Wesley 1975

084 Sullivan, J.E., "Engineering of Quality Software Systems (Measuring the Complexity of Computer Software)", RADC-TR-74-325, Volume V, January 1975 , (A007770).

085 Sullivan, Joseph E., "Simon Prototype Specifications", Working Paper 20144, 24 January 1975

086 Thayer, T.A., "Understanding Software through Empirical Reliability Analysis", Proceedings of 1975 National Computer Conference , May 1975 pp. 335-341

087 Tinanoff, Nathan, et. al. "Structured Programming Series, Volume VI, Programming Support Library (PSL) Program Specifications", 22 November 1974, AD A007 796, RADC-TR-74-300

088 Trainor, W.L., "Avionics Software", NAECON '75 Record, pp. 226-233

089 Trimble, John T., "Structured Programming Series Volume IV, Data Structuring Study", 21 April 1975, RADC-TR-74-300 , (A007796).

090 Trivedi, A.K. and Shooman, M.L. "A Many-State Markov Model for the Estimation and Prediction of Computer Software Performance Parameters", Proceedings International Conference on Reliable Software, 21-23 April 1975, IEEE Catalog No. 75CH0940-7CSR

091 U.S. Air Force Electronics Systems Division, "Summary Notes of a Government/Industry Software Sizing and Costing Workshop", October 1974

092 U.S. Air Force Electronics Systems Division, "Support of AF Automatic Data Processing Requirements through the 1980's (SADPR-85)", Vol. III Technology, Appendix VI, including Annex A, June 1974, AD 783 768

093 Vick, Charles R., "Specifications for Reliable Software", EASCON 1974 7-9 October 1974

094 Voigt, Susan, "Program Design by a Multidisciplinary Team", Proceedings of the 1st National Conference on Software Engineering, September 1975, IEEE Catalog No. 75CH0992-8C

095 Wagoner, W.L., "The Final Report on a Software Reliability Measurement Study", Report No. TOR-0074 (4112)-1, 15 August 1973

096 Wasserman, A.I., "A Top-Down View of Software Engineering", Proceedings of the 1st National Conference on Software Engineering, September 1975, IEEE Catalog No. 75CH0992-8C

097 Williams, P.Y., "DAIS Project Analysis Plan", 18 September 1975, IR-134 (Intermetrics)

098 Williams, R.D., "Managing the Development of Reliable Software", Proceedings of the International Conference on Reliable Software, April 1975, pp. 3-8

099 Willmorth, N.E., "Concepts and Facilities of the SDC Software Factory: Impact", 19 December 1973, TM-5175/820/00

DATA CENTER OPERATIONS

Series 100

101 Berra, P.B. and Mead, L.D., "A Systems Analysis of the Reliability Analysis Center Functions and a Statement of Information System Requirements", IIT Research Institute Report, June 1974

102 Blue, Joseph L., et. al., "Department of Defense Information Analysis Centers", Proceedings of the Meeting of Managers and Users, September 1973

103 COSATI Panel on Information Analysis Centers, "The Management of Information Analysis Centers", Proceedings of a Forum, January 1972

104 Dyke, Freeman H., "How to Manage and Use Technical Information", 1968, Industrial Education Institute

105 Elias, Arthur W., Technical Information Center Administration (TICA) Conference, Drexel Institute of Technology, 1964, Spartan Books, Inc

106 Jacobi, G.J., Lauffenburger, H.A., Llewellen, P.A., et. al., "Development and Implementation of a Reliability Analysis Center", RADC-TR-68-339, July 1968 , (844360).

107 Lauffenburger, H.A., "Annual Report of the Reliability Analysis Center", IIT Research Institute Report, January 1973

108 Lauffenburger, H.A., "1973 Annual Report of the Reliability Analysis Center", IIT Research Institute Report, January 1974

109 Lauffenburger, H.A., "1974 Annual Report of the Reliability Analysis Center", IIT Research Institute Report, January 1975

110 Lauffenburger, H.A., "1975 Annual Report of the Reliability Analysis Center", IIT Research Institute Report, February 1976

111 Lauffenburger, H.A., "Operation of Reliability Analysis Center", RADC TR-71-7, January 1971, (881201).

112 Lauffenburger, H.A., "Reliability Data Banks and Services", NBS/FDA Workshop on Reliability Technology for Cardiac Pacemakers, August 1975, NBS Special Publication 400-28, June 1976

113 Lauffenburger, H.A., Edfors, H. and Peterson, B., "Operation of Reliability Analysis Center", RADC TR-71-304, January 1972, (738649).

114 Little, R.N., "RACIS Alternatives Study", Reliability Analysis Center, IIT Research Institute, November 1975

115 Weinberg, A., "Science, Government, and Information", A Report of the President's Science Advisory Committee, Panel on Science Information, January 1963

116 Wise, R.B., "Options for Data Entry", IIT Research Institute Report, E6330H20-1, October 1975

DATABASE MANAGEMENT AND DATA TRANSLATION

Series 200

201 Bakkom, D.E. and Behymer, J.A., "Implementation of a Prototype Generalized File Translator", International Conference on Management of Data, ACM SIGMOD, May 1975

202 Birss, E.W. and Fry, J.P., "Generalized Software for Translating Data", University of Michigan Technical Report 76DT 3.1, February 1976

203 Boylan, D.L., "DBMS for Minis", Computer Decisions, January 1976 pp. 66-69

204 Capraro, G.T., "A Data Management System Problem Specification Model", Unpublished MS Thesis, Syracuse University, June 1973, RADC-TR-73-193, (766449).

205 CODASYL Data Description Language, Journal of Development, June 1973

206 CODASYL Systems Committee, "Introduction to 'Feature Analysis of Generalized Data Base Management Systems'", Communications of the ACM, Vol. 14, No. 5, May 1971

207 Codd, E.F., "A Relational Model of Data for Large Shared Data Banks", Communications of the ACM, June 1970

208 Codd, E.F., "Recent Investigations in Relational Data Base Systems", Proceedings of IFIP Congress 1974

209 Date, C.J., "An Introduction to Database Systems", Addison-Wesley, 1975

210 Duvall, Lorraine M., "Results and Evaluation of a Functional Test Performed on the Data Base Management System DM-1", Reliability Analysis Center, IIT Research Institute, November 1974

211 Fry, J.P..and Sibley, E.H., "Evolution of Data Base Management Systems", ACM Computing Survey's, March 1976

212 Honeywell Information Systems; Series 60 (Level 66)/6000 Software (GCOS); Management Data Query System/IV Administrator's Guide; DD94; March 1975

213 Honeywell Information Systems; Series 60 (Level 66)/6000 Software (GCOS); Management Data Query System/IV User's Guide; DD92; March 1975

214 Honeywell Information Systems; Series 600/6000 Software, I-D-S Data Query System User's Guide; DC53; April 1974

215 Honeywell Information Systems; Series 600/6000 Software, Integrated Data Store; BR69, Rev. 1; December 1971

216 Honeywell Information Systems; Multics Integrated Data Store Reference Manual (Draft), June 1976

217 Honeywell Information Systems; Multics Relational Data Store Reference Manual (Draft), June 1976

218 Massachusetts Institute of Technology; JANUS Beginner's Manual; 14 July 1975 (Draft)

219 Massachusetts Institute of Technology; JANUS User's Manual; 21 November 1975 (Draft)

220 Muhlhauser, Robert R., et. al., "DM-1 Implementation", Auerbach Corporation, AD 761 520, March 1973

221 Ramirez, J.A., Rin, N.A., and Prywes, N.S., "Automatic Generation of Data Conversion Programs Using a Data Description Language", Workshop on Data Description, Access, and Control, ACM SIGMOD, May 1974

222 Shoshani, Arie, "A Logical Level Approach to Data Base Conversion", International Conference on Management of Data, ACM SIGMOD, May 1975

223 Stamen, Jeffrey P., and Wallace, R.M., "JANUS: A Data Management and Analysis System for the Behavioral Sciences", 1973 ACM Conference, August 1973

224 Su, S.Y.W. and Lam, H., "A Semi-Automatic Data Base Translation System for Achieving Data Sharing in a Network Environment", Workshop on Data Description, Access, and Control, ACM SIGMOD, May 1974

225 Winkler, A.J., et.al., "The Data Administrator's Handbook", January 1976, USAFA-TR-76-1

226 Winkler, A.J., Tufts, R.J., Egel, R.W. and Catalano, J., "Air Force Data Base Management Systems Reference List and Annotated Bibliography", July 1974, USAFA-TR-74-10

227 Winters, E.W. and Dickey, A.F., "A Business Application of Data Translation", International Conference on Management of Data, 1976 SIGMOD, June 1976

228 Yamaguchi, K. and Merten, A.G., "Methodology for Transferring Programs and Data", Workshop on Data Description, Access, and Control, ACM SIGMOD, May 1974

229 Yormark, Beatrice and Stewart, David H., "A Data Management System Evaluation for the Health Insurance Study", Rand Corporation, November 1973, AD 786 593

230 ---, "A Buyer's Guide to Data Base Management Systems", Datapro Research Corporation, November 1974

COMPUTER NETWORKS AND SECURITY

Series 300

301 ---, "Computer Networks: A Tutorial", Revised 1975, IEEE JH3100-5C

302 ---, "Computer Networks: Trends and Applications", Proceedings of the 1974 Symposium, IEEE 74CH0835-9C

303 Cotton, Ira, W., "Network Management Summary", National Bureau of Standards Technical Note 805, February 1974

304 Moore, K.R., "Management Strategies for ADP Networking", Army Computer Systems Command, 1974, AD 785 876

305 Ombudsman Committee on Privacy, The, ACM Los Angeles Chapter, "Privacy, Security, and the Information Processing Industry", 1976

306 Stanford Research Institute, "ARPA Network Current Network Protocols", December 1974, AD/A 003 890

307 Stanford Research Institute, "ARPANET Directory", Network Information Center, NIC 32992, July 1975 (updated)

308 Smith, L., "Architectures for Secure Computing Systems", April 1975, AD A009 221

309 TRW Systems Group, "Computer Security Technology Reference Manual", Rome Air Development Center, December 1974

BIBLIOGRAPHIES

Series 400

401 ---, "Keyword Index", Proceedings of the International Conference on Reliable Software, April 1975, pp. 552-566

402 Duvall, Lorraine, M., "A Software Engineering Bibliography", IITRI E6330-11, May 1976 (190 Entries)

403 Rault, J.-C., Thomson - CSF, January 1976

a. Bibliographies Sur Les Schemas De Programmes (105 Entries)
b. A Bibliography on Program Testing and Verification (1172 Entries)
c. Bibliographie Sur Les Qutils D'Aide Au Test Et a La Mis Au Point Des Programmes (563 Entries)
d. A Bibliography on Computer System Diagnosis (155 Entries)
e. A Bibliography on Software Engineering (609 Entries)

404 Revens, Lee and Neumeyer, Mary, "Bibliography and Subject Index of Current Computing Literature", ACM Computing Reviews, 1974 (1945 Entries)

Appendix A

GLOSSARY

OF

DATABASE MANAGEMENT TERMS

DATABASE MANAGEMENT TERMS

ACCESS

The ability to approach, communicate with (input to or receive output from), or otherwise make use of any resource of an automated data processing system.

ACCESS CONTROL

The process of limiting access to the resources of an automated data processing system to only the authorized persons or (in computer networks) other data processing systems. Access control may be implemented by use of hardware devices, software routines, operating procedures, people, and various combinations of these.

ACCESS TYPE

The kinds of access which may be authorized; e.g., read, write, append, modify, delete, create.

CONTROLLED ACCESS

The property of a software security system that allows each properly identified user to access information and resources within the system to which he or she is authorized, but no more.

DATABASE

a) A generalized, common, integrated collection of data which fulfills the data requirements of all applications which access it, and which is structured to model the natural data relationships which exist in an enterprise.

b) A collection of interrelated data items processable by one or more programs.

DATABASE ADMINISTRATOR

A person or persons given the responsibility for the definition, organization, protection and efficiency of the database for an enterprise.

DATABASE DIRECTORY

A collection of descriptors of all units of data that are available to the database management system, these descriptors being derived from the data description language statements.

DATA CLASSIFICATION

The level of security classification assigned to data.

DATA DESCRIPTION LANGUAGE (DDL)

The language used to describe the database, or that part of the database known to a program.

DATA-ITEM OR ATTRIBUTE

The smallest unit of named data, an occurrence of which is a representation of a value.

DATA INDEPENDENCE

a) The concept of separating the definitions of logical and physical data, such that application programs need not be dependent on where or how physical units of data are stored.

b) The ability to alter the content, format, structure and physical characteristics of a database with a minimal impact on existing programs.

DATA INTEGRITY

The concept that all units of data must be protected against accidental or deliberate invalidation.

DATABASE MANAGEMENT SYSTEM

An integrated set of computer procedures, designed to facilitate storage and retrieval of large amounts of structured data in an orderly fashion, by providing frequently used functions to a user at a level not requiring detailed knowledge of the data structure or major portions of the processing system.

DATA MANIPULATION LANGUAGE (DML)

The language used to cause data to be transferred between the application program and the database.

DATA STRUCTURE

The logical relationships which exist among the units of data in a database and which are under control of a database management system.

ENTITY

A person, place, thing or event of interest to the enterprise, and about which data may be recorded.

ENTITY NAME

The symbol by which a person, place, thing, class or any object of thought is known.

FIELD

A specific defined area of a record used for a particular category of data. A group of character positions used to hold a value.

FILE

A collection of all occurrences of a given type of logical record.

FILE MANAGEMENT

A generic term for the functions of creation, insertion, deletion and modification of records; reorganizing, sorting, merging and other processes commonly performed on files.

HIERARCHICAL STRUCTURE

A method of conceptualizing and/or storing fields, records, or files so that more than one-for-one relationship is defined as existing for a file, and the actual distinctions among hierarchical levels are expressed in the data.

HOST LANGUAGE SYSTEM

A database management system that is built upon the facilities of a programming language and is identified to the application programmer as new tools of the language for logical and physical file manipulations. The tools are embedded in the host language (e.g., FORTRAN, JOVIAL, COBOL) and are accessed usually through CALL statements, but sometimes by extensions in the language.

INDEX

An ordered reference list of the contents of a field or fields of a file, together with a reference notation for identification or location of each field and most likely other related fields. For example, an index might list each value of a field and the location of every record containing that value.

INVERTED FILE

A storage organization in which an index is provided for the values of each type of data-item. If an index is provided for every data type, the file is called totally inverted; if an index is provided for only some of the data item types, the file is partially inverted.

LOGICAL RECORD

The collection of fields as a unit of data to be processed, independent of their physical environment. Portions of the same logical record may be located in different physical records.

NETWORK

One or more collections of directed relationships between three or more units of data such that some units of data are considered owners which others are members, where each member may have more than one owner.

OCCURRENCE

A specific instance of the value of a unit of data.

PASSWORD

A secret word or a string of characters which either identifies or authenticates a user or a defined system resource, such as a program or data.

PHYSICAL RECORD

The collection of fields as a unit of data which is stored, retrieved, or moved as one unit. The unit of data is defined in terms of its physical storage qualities.

PHYSICAL SEQUENCE

An order of records or fields in a file that is the same order in which the records or fields are located in adjacent physical storage.

RECORD

a) A collection of related values treated as a logical unit during any operation of the data management system (e.g., during data collection, processing, or output). (Logical Record)

b) A unit of data to be placed on, or taken from, a storage device in a single operation. (Physical Record)

RECOVERY PROCEDURE

The actions to be taken in restoring a system's computational capability or the data files after any system failure or penetration.

RELATIONAL MODEL

A logical model of data in which records of a given type have a given number of (nonrepeating) items. All instances of a record type are considered to be collected together in a mathematical set. The database is considered to be a collection of time varying relations of varying degrees.

REORGANIZATION

The process of rearranging the relative physical placement of data-units in the database.

REPEATING GROUP

A set of associated fields in a record or in another repeating group which may have more than one value. Individual values for one field are associated with individual values in the other fields (which may include the null value). A repeating group containing only one field is generally called a multivalued field.

REPORT GENERATOR

A computer program, usually included in a database management system, that can direct the production of formatted output reports if it is provided with format specifications, input file detail, and other information.

RESTRUCTURE

The process of adding to or deleting from the types of data-units and data-relationships represented in the database, of rearranging data-units which are components of larger data-units, and of making the corresponding changes to its schema.

RETRIEVAL

The operation in a database management system used to recover data stored in a database in such a way that it is available to other database management system functions. A retrieval generally involves conditional selection of data and the assembling of selected records.

RETRIEVAL SPECIFICATIONS

The manner and syntax of specifications used to express the conditions under which files, records, or fields in a database will be selected for further processing during a retrieval operation.

ROLL-BACK

The process of reversing recent activities of the system, to restore some of or all of the database to its state at a previous point in time.

SCHEMA

A complete description of the database in terms of the characteristics of the data and the implicit and explicit relationship between data-units.

SECURE OPERATING SYSTEM

An operating system effectively utilizing those hardware and software control functions necessary to provide the protection appropriate for the value of the information and resources managed by the system.

SECURITY

In the computer community, the realization of protection for hardware, software, and data.

a) Administrative Security -- The management constraints, operational procedures, accountability procedures, and supplemental controls used to provide an acceptable level of protection for secure material

b) Computer Security -- The hardware/software functions, characteristics and features needed to provide an acceptable level of protection in a computer system.

c) Data Security -- The protection of data from accidental or intentional but unauthorized modification, destruction, or disclosure.

d) Information Security -- The hardware/software functions, characteristics, and features; operation procedures, accountability procedures, and access controls at the center computer facility, remote computer and terminal facilities; the management constraints, physical structures, and devices; and personnel and communications controls needed to provide an acceptable level of protection in a computer system.

e) Personnel Security -- Insuring that all personnel who have access to any sensitive data have the appropriate clearances.

f) Physical Security -- Physical security, as it pertains to computers, does not differ from physical security for other installations. It is achieved through the use of locks, guards, badges, personnel security clearances and administratively controlled measure outside the computer as well as measures required for the protection of the structures housing the computer and related equipment against damage from accident, fire and environmental hazard, thus ensuring the protection of their contents.

SELECTED RECORDS

The set of complete or partial records which are the result of a retrieval operation and become available for further processing, such as for sorting and printing.

SELECTION CRITERIA

The conditions expressed or implied by a user which are used selectively to recover data during a retrieval operation; i.e., a logical expression.

SELF-CONTAINED SYSTEM

A database management system, the capabilities and language of which are intended primarily for the nonprogrammer. They are self-contained in the sense that they usually have no connection with any procedural language (except that the system itself may be written in a procedural language, or it may permit user-written code in a procedural language).

SENSITIVE INFORMATION

Information whose disclosure or modification is disadvantageous.

SEQUENTIAL FILE STRUCTURE

A method of storing and retrieving data in which information becomes available in a one-after-the-other sequence only.

SET

A named collection of related records representing a one-to-many relationship between the owner and member records.

SORT KEY

A field of a record whose values will be used in arranging the records, or just the values, into a sorted sequence.

STORAGE AND RETRIEVAL SYSTEM

A system for storing and locating, on demand, certain documents or other records relevant to a given information requirement from a file of such material. Examples are classification, indexing, and machine searching systems.

SUBSCHEMA

A description of those data-units and relationships from a database of interest to a particular program.

SUBSET FILE (also Data Subset)

A selected portion of a data file arranged so that it is also (or can be used as if it were) a data file, separately processable by the database management system.

Appendix B

SURVEY RESULTS

Appendix B

SURVEY RESULTS

ABSTRACT

The results of a survey on the exchange of software development experience information are presented in this Appendix. These results indicate that members of the software community are interested in the establishment of a center for information sharing and are willing to share their experiences and data on developing computer software. They are interested in the results of analyses on determining the effects that different developmental and testing philosophies have on the cost and productivity of software development.

Table of Contents

List of Figures

List of Tables

1. Introduction

To develop the specifications for the Software Data Repository, it was necessary to determine the potential users of the data center and the software engineering areas in which they are interested. To ascertain the types of potential users and their technological interest, we performed a survey of a portion of the software development community.

Because a software experience database would be the heart of the data repository, the survey also contained questions concerning the data collection efforts of the recipients and their opinions on sharing their data and their experiences in collecting the data.

This appendix is organized as follows. First, the background for the survey is presented followed by an initial analysis of the survey. Then, the results of further *analysis is presented*, correlating response categories.

2. Survey Approach

During the summer of 1975, a survey was sent to selected members of the software development community. A total of 1659 copies of the survey were mailed. Two mailing lists were used:

1. Recipients of the RADC-sponsored "Structured Programming Series" publications RADC-TR-74-300 (875 names).

2. Attendees of the International Conference on Reliable Software April 1975 (784 names)

By the end of October, 337 responses had been received -- a response rate of 22%.

The response card used for this survey is reproduced here as Figure 1. The questions on this survey card contain four major areas dealing with:

A. Usage level of the repository

B. Data collection activities and cooperation posture

C. The characteristics of the respondees

D. Interest level for the repository

3. Survey Results

The survey results are discussed below relative to each of the questions contained on the survey form.*

* For a more detailed analysis of the results, see Duvall, L.M., "Software Data Repository Study - Survey Results", IITRI Report E6330-5, February 1976

Usage

The survey question to determine the usage interest was in matrix form. The rows designated a technology area (Reliability Prediction, Testing Techniques, Development tools, Other) and the columns designated a usage level (Data for your own Analysis, Reports on Analysis Tools, Results of Analysis). Contained in Table I is a summary of these responses. In the first three columns, the numbers represent the number of times that cell was checked; the "% of Total" represents the percentage of survey cards that contained a check mark in that cell. For example, 136 survey cards contained a check mark for the cell "Reliability Prediction, Data for Your Own Analysis". Of all cells, this cell was checked the least number of times whereas "Testing Techniques, Results of Analysis" was checked most often (226 times).

Figure 2 contains histograms representing the percentage of respondees that checked each individual cell. In all three technology areas, "Data for Your Own Analysis" was checked the least number of times (40% for Reliability Prediction, 47% for Testing Techniques, and 48% for Development tools), and "Results of Analysis" was checked the greatest number of times (65% for Reliability Prediction, 67% for Testing Techniques and 66% for Development Tools).

As shown in Figure 3, 75% checked one or more cells in the "Reliability Prediction" row, 84% checked one or more cells in the "Testing Techniques" row and 81% checked one or more cells in the "Development Tools" row.

Thirty-four percent of those that responded included comments in the "Other" row. The keywords most often mentioned or implied here were: cost, management, reliability, productivity development and production design languages, testing standards, and maintainability.

Data Collection

Sixty-two percent of the respondees indicated they were collecting data (Figure 4), 77% indicated they would be willing to share experiences (Figure 5), and 76% indicated they would support making the data available (Figure 6).

Of those respondees that indicated they were collecting data, 83% stated they would be willing to share experiences and 82% indicated they would support making the data available (Table II). Of those that were not collecting data, 68% noted that they would share the data (i.e. the data they did not have). This anomoly most likely occurred because the respondees wanted to indicate that they would be willing to share if they did have the data.

Seventy-five comments were made in this section even though there was no specific "other" category. Under the collection question, the majority (70%) of the comments were concerned with qualifying their collection (e.g. planning, informally, some, etc.). In addition, six specifically noted cost data.

For the other two questions on sharing experiences and data, the majority (75%) of the comments had to do with the need for approval and proprietary limitations.

User Attributes

Of those that responded, 25% stated they were from the government, 43% from industry, 20% from universities and 12% "other" (Figure 7). The "other" category means that either none or more than one box was checked. If more than one box was checked, the respondee is most likely teaching and consulting for the government or industry, or working for an industrial firm on government contracts.

To provide a comparison of those who responded to those who were recipients of the survey, the mailing lists were analyzed by categorizing the institutional type of the recipients (Table III). Most of the recipients were not difficult to categorized except for the "other" category. As can be seen, the number that responded followed closely to those who received the survey.

The survey also asked the potential users to indicate their principal functions as: design software, software development research, and/or evaluate software. The highest percentage (64%)of respondees noted that one of their principal functions was designing software; the lowest percentage (41%) indicated a user of software (Figure 8).

Receiving/Providing Further Information

The last question on the survey was included to provide us with some indication as to interest in the repository. The response was overwhelmingly positive. Of those that responded, 307 (or 91%) indicated "yes" to the question: "Are you interested in receiving and/or providing further information?", 13 (or 4%) indicated "no", and 17 (5%) checked neither box.

Usage vs Institution Type

The data on institution type as related to potential usage showed that the respondees of the universities were slightly more interested in development tool effects than those from the government or industry. Those from the government expressed more interest in obtaining the results of analyses than those from industry or the universities. Table IV contains summary data correlating this potential usage of the data repository and the Institution Type. For example, 36 respondees indicated they were from the government and were interested in "Reliability Prediction, Data for Your Own Analysis". Referring to Table I, 36 of the 136 who indicated interest in "Reliability Prediction, Data for Your Own Analysis" were from the government. In reviewing this column, 123 cells out of a total of 490 cells (25%) checked by the government respondees indicated interest in data for some technology area. This is the lowest percentage for any category vs institution type. However, there is little variation when comparing the total percentages to the individual percentages for each institution type.

Data Collection vs Institution Type

Within institutions, the greatest percentage (69%) collecting data are from the government; the least percentage (48%) collecting data are from the universities; and the respondees from the universities are the most willing to share experience and data (88% and 85%, respectively). Although still an encouraging response, those from industry were least willing to share experiences (70%). The distribution of the respondees by institution type to those responding "yes" to the data collection question is illustrated in Figure 9; to those responding "yes" to the sharing experience question is shown in Figure 10; and to those responding "yes" to the sharing data question is illustrated in Figure 11.

Function vs. Institution Type

The highest percentage of respondees that stated they were designing software was from industry (72%) and the highest percentage of respondees for implementing software (69%) was also from industry. The government respondees showed a slight edge over industry (49% vs. 46%) for evaluating software; the universities showed a slight edge over industry (44% vs 42%) as users of software (Figure 12).

The largest differential occurred in the software development research area. The universities indicated 71% compared to 61% for "other", 50% for industry and 49% for government.

Interest vs Institution Type

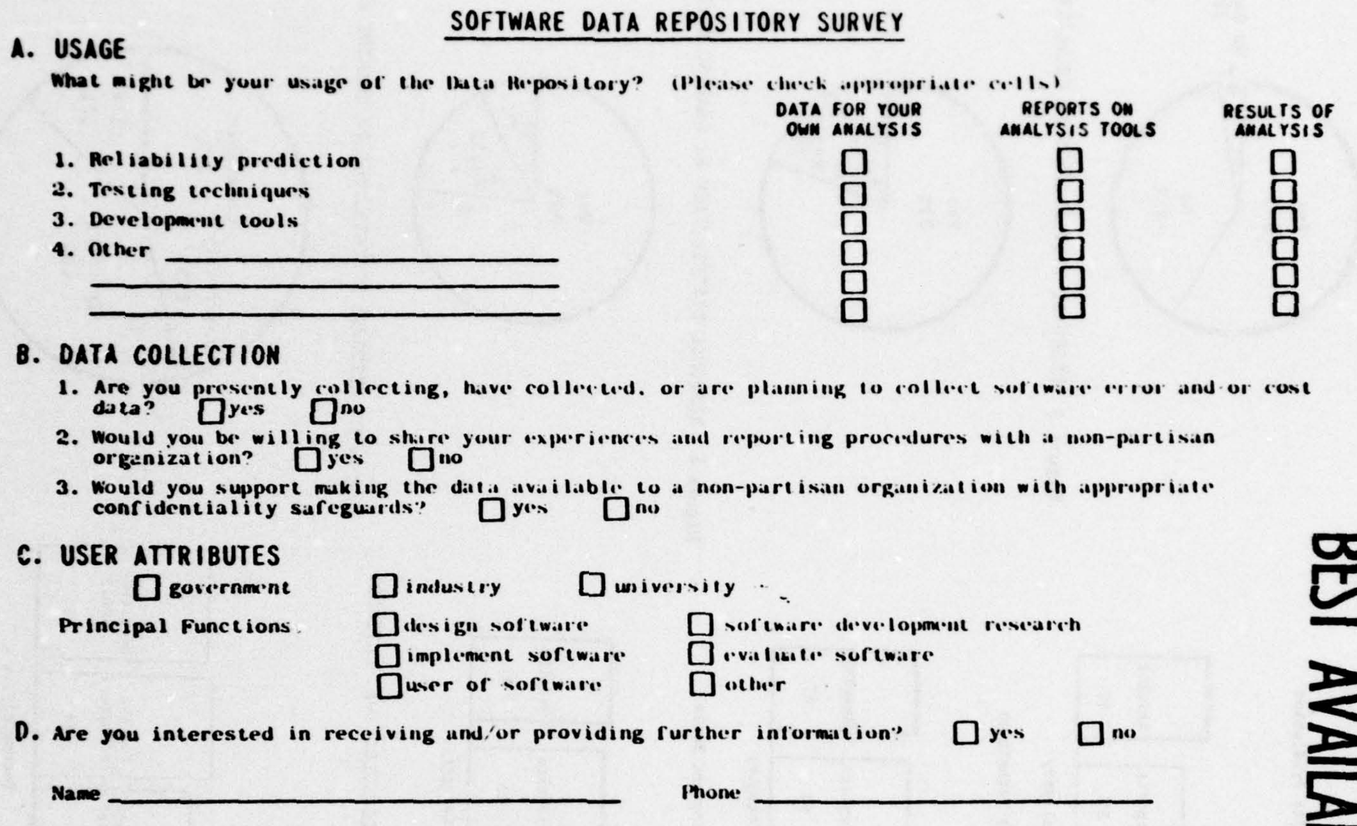
As seen in Figure 13, 91% of all the respondees indicated they were interested in receiving/providing further information. The highest percentage of affirmative responses came from industry (94%). The lowest percentage of negative responses came from government (2%).

4. Conclusions

First and foremost, the survey results indicate that the software development community is interested in the establishment of a software data repository. The potential users are most interested in receiving information on the effects that different tools and techniques have on the software development process; and, secondarily, are interested in having access to data so that they can perform their own analyses. The effects on productivity, costs and reliability are also areas of user interest.

Most of the potential users are either collecting data or planning to do so, and an overwhelming majority are willing to share these data with a nonpartisan organization. The user group showing the highest level of data collecting activities is the government, rather than industry or the universities. University personnel are the most willing of the three user groups to share their data.

The principal functions of government and industrial users are designing and implementing software; whereas, with the universities, the principal function is performing research in the area of software development.



SOFTWARE DATA REPOSITORY SURVEY

A. USAGE

What might be your usage of the Data Repository? (Please check appropriate cells)

	DATA FOR YOUR OWN ANALYSIS	REPORTS ON ANALYSIS TOOLS	RESULTS OF ANALYSIS
1. Reliability prediction	☐	☐	☐
2. Testing techniques	☐	☐	☐
3. Development tools	☐	☐	☐
4. Other ______	☐	☐	☐
______	☐	☐	☐
______	☐	☐	☐

B. DATA COLLECTION

1. Are you presently collecting, have collected, or are planning to collect software error and/or cost data? ☐ yes ☐ no
2. Would you be willing to share your experiences and reporting procedures with a non-partisan organization? ☐ yes ☐ no
3. Would you support making the data available to a non-partisan organization with appropriate confidentiality safeguards? ☐ yes ☐ no

C. USER ATTRIBUTES

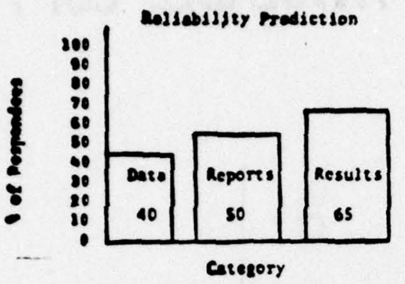
☐ government ☐ industry ☐ university

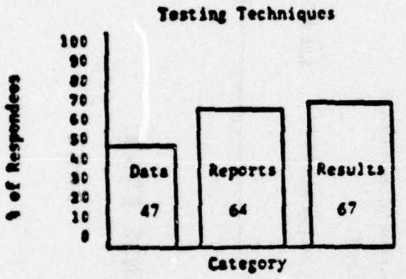
Principal Functions. ☐ design software ☐ software development research
☐ implement software ☐ evaluate software
☐ user of software ☐ other

D. Are you interested in receiving and/or providing further information? ☐ yes ☐ no

Name ______ Phone ______

Figure 1 SURVEY RESPONSE CARD





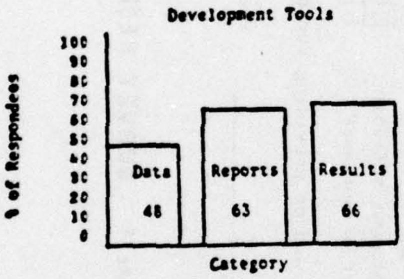


Figure 2 TECHNOLOGY vs. CATEGORY

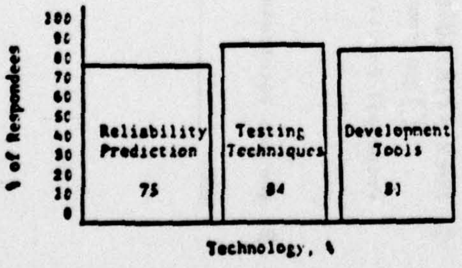


Figure 3 RESPONSE DISTRIBUTION BY TECHNOLOGY

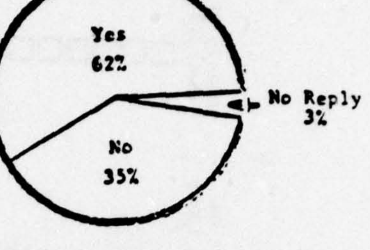


Figure 4 RESPONSE DISTRIBUTION BY COLLECTING DATA

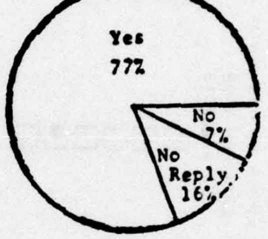


Figure 5 RESPONSE DISTRIBUTION BY SHARING EXPERIENCE

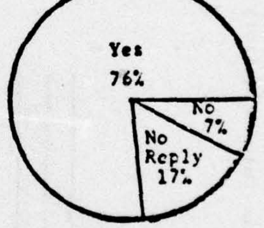


Figure 6 RESPONSE DISTRIBUTION BY SHARING DATA

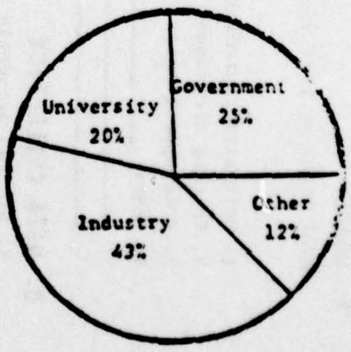


Figure 7 RESPONSE DISTRIBUTION BY INSTITUTION TYPE

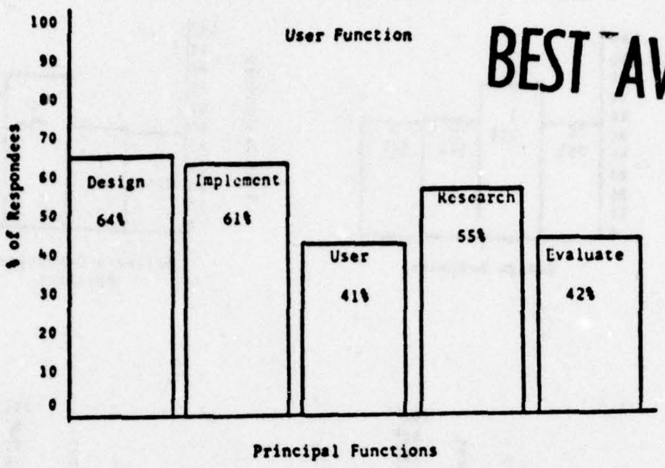


Figure 8 RESPONSE DISTRIBUTION BY PRINCIPAL FUNCTIONS

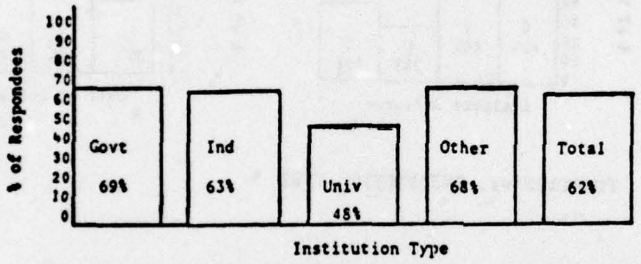


Figure 9 YES, COLLECTING DATA vs. INSTITUTION TYPE

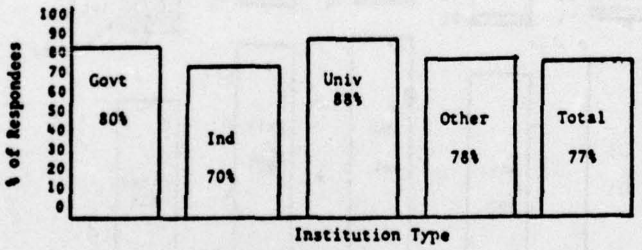


Figure 10 YES, SHARING EXPERIENCES vs. INSTITUTION TYPE

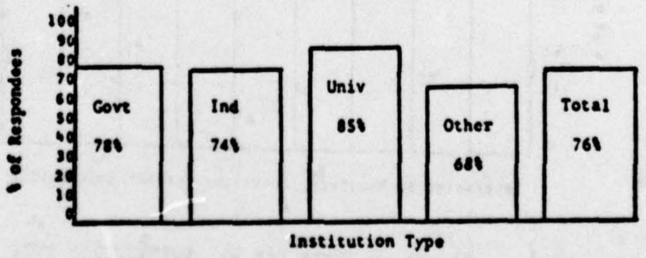


Figure 11 YES, SHARING DATA vs. INSTITUTION TYPE

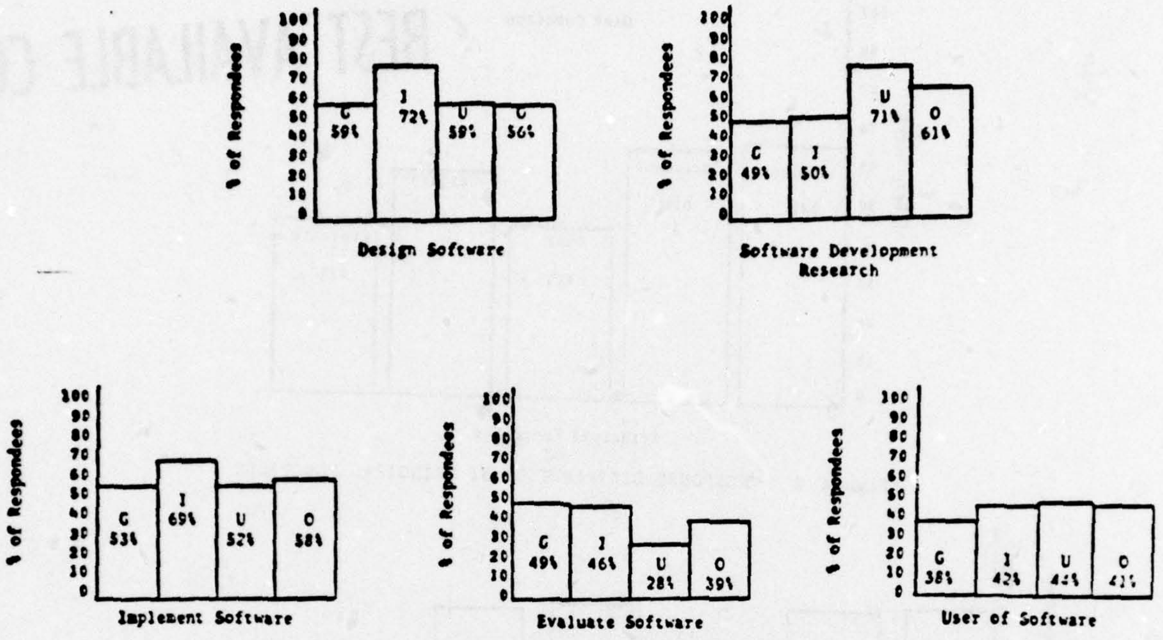


Figure 12 FUNCTION vs. INSTITUTION TYPE

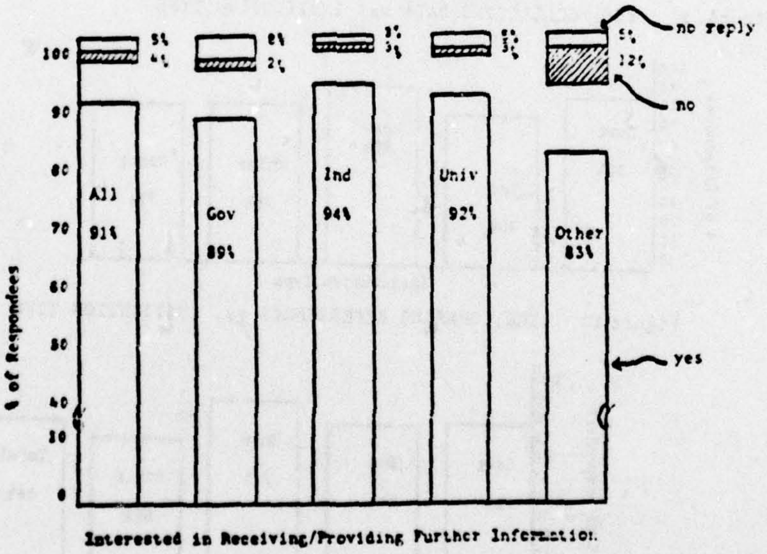


Figure 13 INTEREST vs. INSTITUTION TYPE

Table I

USAGE SUMMARY TABLE

Technology	Data for Your Own Analysis		Reports on Analysis Tools		Results of Analysis		TOTAL OCCURRENCES EACH TECHNOLOGY	
	No.	% of Total*	No.	% of Total*	No.	% of Total*	No.	%
Reliability Prediction	136	(40)	169	(50)	218	(65)	523	(30)
Testing Techniques	159	(47)	216	(64)	226	(67)	601	(35)
Development Tools	160	(48)	211	(63)	223	(66)	594	(35)
ALL TECHNOLOGIES TOTAL OCCURRENCES	455	(26)	596	(35)	667	(39)	1718	(100)

*Total number of responses = 337.

Table II COLLECTING DATA vs. SHARING EXPERIENCES AND DATA

Yes, Share Experience, %		Yes, Share Data, %
68	not collecting data	68
50	no reply	50
83	yes, collecting data	82

Table III

INSTITUTION TYPE - RECIPIENTS vs. RESPONDEES

Institution Type	Mailed, %			Responses (both)	
	1st	2nd	Both	%	No.
Government	41	11	27	25	85
Industry	48	63	55	43	146
University	9	24	16	20	65
Other	2	2	2	12	41
Total	100	100	100	100	337

Table IV

SUMMARY: USAGE vs. INSTITUTION TYPE

		Data for Your Own Analysis				Reports on Analysis Tools				Results of Analysis				Technology Totals							
														G		I		U		O	
Technology		G	I	U	O	G	I	U	O	G	I	U	O	No.	%	No.	%	No.	%	No.	%
Reliability Prediction		36	62	18	20	44	79	26	20	63	97	34	24	143	29	238	32	78	27	64	35
Testing Techniques		45	71	27	16	63	95	36	22	72	94	37	23	180	37	260	34	100	35	61	33
Developmental Tools		42	71	31	16	57	94	39	21	68	94	39	22	167	34	259	34	109	38	59	32
Totals	G	123 (25%)				164 (33%)				203 (41%)				490 (100%)							
	I		204 (27%)				268 (35%)				285 (38%)					757 (100%)					
	U			76 (27%)				101 (35%)				110 (38%)						287 (100%)			
	O				52 (28%)				63 (34%)				69 (38%)							184 (100%)	

Key:
G - Government
I - Industry
U - University
O - Other

Appendix C

SOFTWARE ENGINEERING TERMS FROM THE LITERATURE

Appendix C

SOFTWARE ENGINEERING TERMS FROM THE LITERATURE

1. Introduction

This Appendix contains definitions of software engineering terms as they appear in the literature. A list of the papers and reports that were searched and used for the term definitions is included in Section 2, along with the corresponding reference number.

Section 3 contains a list of the terms, the reference number (in parenthesis) of the document that contained the definition, and the definition.

2. References

001 TRW Systems Group, "Software Reliability Study", RADC-TR-74-250, October 1974, AD 787 784

002 Shooman, Martin L., "Software Reliability: Measurement and Models", Proceedings 1975 Annual Reliability and Maintainability Symposium, Jan. 1975, pg. 485-491

003 Reifer, D.J., "Interim Report on the Aids Inventory Project", Report SAMSO-TR-75-184, 16 July 1975

004 Trimble, John T., "Structured Programming Series Volume IV, Data Structuring Study", RADC-TR-74-300, 21 April 1975, (A015794).

005 Barry, Barbara S., "Structured Programming Series, Volume X, Chief Programmer Team Operations Description", RADC-TR-74-300, 22 January 1975, AD A008 861

006 Smith, Ronald L., "Structured Programming Series, Volume IX, Management Data Collection and Reporting", RADC-TR-74-300, 24 October 1974, AD A008 640

007 Kessler, Marvin M., Kister, William E., "Structured Programming Series Volume XIV, Software Tool Impact", RADC-TR-74-300, 22 May 1975, (A015795).

008 Kraly, Thomas M., et. al., "Structured Programming Series, Volume VIII, Program Design Study", RADC-TR-74-300, 30 May 1975, (A016415).

009 Smith, Ronald L., "Structured Programming Series, Volume XV, Validation and Verification Study", RADC-TR-74-300, May 1975, (A016668).

010 Manley, John H., "Embedded Computer System Software Reliability", Defense Management Journal, Vol. 11, No. 4, October 1975, pgs. 13-17

011 Wagoner, W.L., "The Final Report on a Software Reliability Measurement Study", Report No. TOR-0074(4112)-1, 15 August 1973, Contract No. F04701-73-C-0074

012 Bratman et. al., "The SDC Software Factory", 1 August 1973, TM-5075/100/00

013 Ross, Douglas, T., Goodenough, John B., Irvine, C.A., "Software Engineering: Process, Principles and Goals", Computer, May 1975

014 Schwartz, Jules I., "Construction of Software: Problems and Practicalities", Practical Strategies for Developing Large Software Systems, Ellis Horowitz, Ed., Addison Wesley 1975

015 Boehm, Barry W., "Software Design and Structuring", Practical Strategies for Developing Large Software Systems, Ellis Horowitz Ed., Addison-Wesley 1975

016 Boehm, B.W., et. al., "Characteristics of Software Quality", TRW-SS-73-09, 28 Dec 1973

017 Clapp, J.A., "Software Engineering: Problems and Future Developments", Nov. 1974, MTR-2791, ESD-TR-74-195

018 Luppino, F.M. et. al., "Structured Programming Series, Volume V, Programming Support Library (PSL) Functional Requirements", RADC-TR-74-300, 25 July 1974, AD/A 003 339

Definitions

Accreditation (012)
All activities that, taken together, establish a sufficient level of confidence in the final product that the developer is able to guarantee its functional performance to specifications and to provide a warranty to the customer with minimum risk of additional support at the developer's expense.

Accuracy study processor (003)
A computer program used to perform calculations to assist in determining if program variables are computed with required accuracy.

Actual data (018)
Data describing the results of programming actions for a project that will be the primary data included in the management reports.

Analytical Modeling (003)
The the technique used to express mathematically (usually by a set of equations a representation of some real problem. Such models are valuable for abstracting the essence of the subject of inquiry. Because equations describing complex systems tend to become complicated and often impossible to formulate, it is usually necessary to make simplifying assumptions which

may distort accuracy. specific language and simulation systems serve as aids to implementation.

Analyzer (003)
A computer program used to provide source language or execution frequency statistics at the program or source-statement level to assist in performance evaluation and determination of test case coverage.

Assemblers (012)
Tools that translate programs written in symbolic machine language into actual machine language programs.

Automated network or path analysis (009)
A technique which defines a practical measurable means by examining source code and determining the minimum set of paths which exercise all logical branches of a program.

Automated test generator (003)
A computer program that accepts inputs specifying a test scenario in some special language, generates the exact computer inputs, and determines the expected results.

Automated verification systems (003)
Computer programs that instrument the source code by generating and inserting counters at strategic points to provide measures of test effectiveness. They provide data that details how thoroughly the source code has been exercised.

Block diagram (005)
A diagram of a system, instrument, or computer, in which the principle parts are represented by suitability associated geometrical figures to show both the basic functions and the functional relationships among the parts. (ANSI)

Budgeting and estimating (012)
Those activities that determine the levels of effort and resources needed to accomplish a project.

Bug (002)
one or more software bugs exist in a system if a software change is required to correct a single major error or minor error so as to meet specified or implied system performance requirements.

Certification (009)
Carries the connotation of an authoritative endorsement and

seems to imply testifying in writing that the program is of a certain standard or quality. Certification usually implies the existence an
independent quality control group. Certification extends the process of verification and validation to an operational environment and involves acceptance testing of the overall system.

Certification (012)
The process of demonstrating that the system meets its customer requirements and guaranteeing that compliance in writing.

Certification tools (012)
Information reporting and summarizing components that provide a systematic data collection and summarizing mechanism that produces a system status report.

Change (002)
Any alteration (addition, deletion, correction) of the program code whether it be a single character or thousands of lines of code. Changes made to improve documentation or satisfy new specifications are important to record and study but are not counted as bugs.

Chief programmer team (017)
This concept implies the use of a highly structured team of specialists for software production relying on technical procedures which are based on structured programming principles, and office procedures backed up with automated aids to support group communication. Specialized roles for people on a project, and the relationships among them, are well-defined.

Comparator (003)
A computer program used to compare two versions of the same computer program under test to establish identical configuration or to specifically identify changes in the source coding between the two versions.

Compiler (012)
A tool, used in the production of software systems, that allows programs to be written in higher-order languages. Examples include: PL/I compiler, FORTRAN Compiler, and COBOL compiler.

Computer data (010)
A representation of facts, concepts or instructions in a structured form suitable for acceptance, interpretation

or processing by communication between computer equipment. Such data can be external (in computer-readable form) or resident within the computer equipment and can be in the form of analog or digital signals.

Computer equipment computer hardware (010)
Devices capable of accepting and storing computer data, executing a systematic sequence of operations on computer data or producing computer outputs. Such devices can perform substantial interpretation, computation, communication, control and other logical functions. Examples: central processing units, terminals, printers, analog/digital converters, tape drives, disks and drums.

Computer program certification (003)
The test and evaluation of the complete computer program aimed at ensuring operational effectiveness and suitability with respect to mission requirements under realistic operating conditions.

Computer program validation (003)
The test and evaluation of the complete program aimed at ensuring compliance with performance and design criteria.

Computer program verification (003)
The iterative process of determining whether the product of each step of the computer program acquisition process fulfills all requirements levied by the previous step.

Computer software (010)
A combination of associated computer programs and data

required to command the computer equipment to perform computational or control functions.

Computer system (010)
An interacting assembly consisting of computer equipment computer programs and computer data.

Computer turnaround time (018)
the time differential between the submittal of a job to the Computer Center and the return of the job results to the programmer.

Configuration control (012)
A methodology concerned with procedures for controlling the contents of a software system. A way of monitoring the status of system components, preserving the integrity of releases and developing versions of a software system, and controlling the effects of changes throughout the system.

Configuration management (012)
All activities related to controlling the contents of a software system. It monitors the status of system components, preserves the integrity of released and developing versions of a system, and controls the effects of changes throughout the system. It is a process dealing as much with procedures as with tools.

Constants auto checker (003)
A computer program used to search a tape for all constants and parameters to identify the name of the constant, its storage location, and the binary scale factor. These are then compared with specification values to assure compliance.

Control (012)
A major sub-division without configuration management. The procedures by which changes to the Design Requirements are proposed and formally processed.

Conversion aids (007)
Those software tools which assist in converting operational software from one compiler to another. These tools analyze

the source code as written for one compiler and highlight those statements which are not compatible with the capabilities of the target compiler. In some instances these conversion aids will replace the incompatible statements with one or more target compiler statement which are designed to achieve the same result.

Correctness proof (009)
The technique of proving mathematically that a given program is consistent with a given set of specifications. This process can be accomplished by manual methods or by program verifiers requiring manual intervention.

Correctness proofs (003)
Automated verification systems exist which allow the analyst to prove small programs are correct by means similar to those used in proving mathematical theorems. Axioms and theorems derived are used to establish validity of program assertions and to provide a fundamental understanding of how the program operates.

Cost data (018)
Data describing the costs associated with the resources expended by a software project.

Cost estimation (012)
A standard technique for estimating the amount of labor necessary for the completion of a task, the amount and potential costs of computer time required, etc., prior to and during a project's lifetime.

Cost management (012)
A collected set of tools that provide the criteria and devices for tracking project costs.

Cross compiler (012)
A tool that can operate on a host computer and produce code for a designated external computer.

Cross-assembler (003)
A computer program that accepts symbolic instruction mnemonics for a selected target computer and generates target computer machine code while hosted on another computer. A cross-assembler thus allows code written for one computer to be assembled on another.

Cross-reference program (003)
A group of computer programs that provide cross-reference information on system components. For example, programs can be cross-referenced with other programs, macros, parameter names, etc. This capability is useful in problem-solving and testing to assess impact of changes to one area or another.

Cross-reference tools (007)
Utility programs which provide cross-reference data concerning a program written in a higher level language. These utility programs analyze a source program and provide as output such data as follows:
1. Statement label cross-index
2. Data name cross-index
3. Literal usage cross-index
4. Inter-subroutine CALL cross-index
5. Statistical counts of statement types

Cyclic data (018)
Data on programming activities that occurred since the last management reporting period.

Data (018)
A set of facts

Data analysis tools (012)
Programs specifically designed to perform statistical and comparative analyses on data produced during the execution of a program test.

Data base analyzer (003)
A computer program that reports information on every usage of data, identifies each program using any data elements, and indicates whether the program inputs, uses, modifies, or outputs the data element. Any unused data is printed. Errors dealing with misuse and non-use of data and conflicts in data usage are identified.

Data definition language (003)
A computer program used to describe data at a sufficiently high level in order to make the use of a particular programming language transparent to the data definition process. This language allows us to specify the data so that multiple languages can share and use it.

Data item (018)
A specific entity of data.

Data reduction tools (012)
Often data-base dependent, these programs process a data set and convert the information into readable form. These programs perform statistical and comparative transformations on the recorded data obtained by instrumentation.

Debugging (009)
Debugging starts with known errors and attempts corrections.

Debugging (012)
The testing of a program for proper execution. Includes the detection, diagnosis, recovery, and correction of program errors.

Also see Validation and Debugging Tools.

Debugging tools (012)
Those programs designed to locate and eliminate programming errors and to test a program for proper execution. Also see Validation and Debugging Tools.

Decision table (008)
A tabular representation of the following three items:
1. Conditions - factors to consider in making a decision.
2. Actions - steps to be taken when a certain combination of conditions exist.
3. Rules - specific combinations of conditions and the actions to be taken under those conditions.

Deliverable code indicator (018)
Indicates whether or not the code will be delivered to the customer.

Delivery (001)
The point at which the software package is turned over to a customer for use in the operational environment.

Design implementation specifications (012)
This document defines the system and program design. It includes block and program flow diagrams, set-used information on all data descriptions of all data and may include narrative descriptions of the operation of every program in the system.

Design language (003)
A computer program used to proivde an understandable representation of the software design as it evolves. These programs allow designs to be constructed and expanded in a hierarchical fashion. They document the design and the decisions that led to it.

Design requirements baseline document (012)
This document is the baseline description of the object system; hence, the foundation upon which system design and configuration control proceed.

Design simulation (009)
A technique that describes a proposed system, produces a computer based "model" or simulated system, and then evaluates the effect of various system requirements and design alternatives.

Design validation (009)
The examination or inspection of the functional requirements and the design of a software system for the purpose of finding errors. Other terms used to describe this technique or variations of this technique include design review, project review, design inspections, walk-throughs, and inprocess reviews. This technique is similar to the design verification technique except that it is performed earlier in the software development cycle and at a system functional level.

Design verification (009)
The examination or inspection of a software specification for the purpose of finding design errors. Other terms used in the literature to describe this technique or variations of this technique include design review, design inspection, specification testing, paper testing walk-through, structured walk-through, and preliminary

design review.

Development test and evaluation (003)
Test and evaluation that focuses on the technological and engineering aspects of the system, or equipment items (AFR 80-14).

Diagnostics debug aids (003)
Compile and execution time checkout and debug capabilities that help identify and isolate program errors. These capabilities usually include commands or directives such as DUMP, TRACE, MODIFY, CONTENTS, BREAKPOINT, etc.

Direct interface (001)
An interface immediately between two software elements.

Documentation (012)

the production of all the paper work necessary to describe the final product. Examples include: cross-reference listings, dictionary listings, and flow charts.

Driver program (009)
Superfluous (throw-away) code needed to perform the unit testing and lower levels of integration testing in a bottom up software development effort.

Drivers (009)
Computer based informal testing technique and are used in conjunction with other techniques almost exclusively during the system implementation phase of a software development project.

Dynamic simulator (003)
A computer program used to check out a program in a simulated environment similar to that in which it will reside. Closed-loop effects between computer and environmental models are gained when inputs and outputs are responded to by the various models. The simulator allows the environment to be stablized at a specific configuration for any number of runs required to observe, diagnose, and resolve problems in the operational program.

Editor (003)
A computer program used to analyze source programs for coding errors and to extract information than can be used for checking relationships between sections of code. The editor will scan source code and detect violations to specific programming practices and standards, construct an extensive cross-reference list of all labels, vAriables, and constants, and check for prescribed program formats.

Egoless programming (017)
This term was coined by Weinberg (The Psychology of Computer Programming, Van Nostrand Reinhold Company New York, 1971). He felt that some programmers can become psychologically attached to their programs as extensions of themselves so that errors in programs become damaging to the programmer's self-image. An egoless programming environment is one that creates an attitude that open, shared programming is good. By making code publicly available, the ownership of code is discouraged, and individuals are encouraged to write code that will be clear and understandable to others.

Element (001)
A Grouping of Routines which performs a perscribed function.

Embedded computer system ECS (010)
A computer system that is integral to an electromechanical system such as a combat weapon system; tactical system; aircraft, ship missile, spacecraft, certain command and control systems; and civilian systems such as an automated rapid transit system. Embedded computer systems are primarily differentiated from automatic data processing systems (ADPs) by how they are developed, acquired and operated in a using system.
Its key attributes are:
1. It is physically incorporated into a larger system whose primary function is not data processing.
2. It is integral to such a larger system from a design, procurement and operations viewpoint.
3. Its outputs generally include information, control signals and computer data.

Engineering Scientific Simulations (003)
An engineering simulation is used to study system characteristics, develop algorithms, and provide data that act as a standard for testing. These programs generally simulate subsystems at varying degrees of complexity depending on the subsystem being studied or the use made of the simulation. They generally consist of a set of modules, each of which is assigned a specific simulation function and is designed with well-defined inputs and outputs and precise

interfaces. Each module performs an assigned simulation
function to vary the method, speed of computation,
accuracy, complexity, etc. The structure can encompass all basic
simulation capabilities for simulation of continuous and
discrete systems.

Engineering change proposal (012)
The formal vehicle by which a change in the baseline document
is proposed. It describes the nature and magnitude of a
proposed change and the impact of the change on all elements of
the system.

Environment simulator (003)
A computer program used to permit testing of operational
programs on a host computer. The operational programs
run under simulated conditions as if they were operating within the
real-time control program of a machine to which all of the devices
constituting the ultimate system are attached. The simulator
program contains expansions of all control program macros that
modify the entry block and other working storage in the same manner
as the macros in the actual control programs.

Equate program (003)
A computer program that lists all equivalences found in
examined code and prints warnings when multiple equivalences
are found.

Estimating (012)
Determining what levels of effort and what resources need
to be applied to accomplish the desired results. Also see
Budgeting and Estimating.

Execution analysis (009)
The automated monitoring of the computer based software testing
activities, collection data from these testing activities,
and subsequently predicting, by manually analyzing the data,
the duration and cost of testing, and the quality of the software
product. Other terms used in the literature referring to this technique
or variations of this technique include code analyzer, code auditor,
program evaluator, and product assurance evaluator.

Flight tests (003)
A technique used to demonstrate hardware and software
performance in actual system operation.

Flow charts (012)
Diagrams of a program's logic flow.

Flowchart (003)
A graphical representation for the definition, analysis, or
solution of a problem, in which symbols are used to
represent operations, data, flow, equipment, etc. (ANSI).

Flowcharter (003)
A computer program used to show in detail the logical
structure of a computer program. The flow is determined
from the actual operations as specified by the
executable statements, not from comments. The flowcharts
generated can be compared to flowcharts provided in the
computer program specification to show discrepancies
and illuminate differences.

Flowcharting tools (007)
Utility programs which atuomatically draw program flowcharts
directly from source code.

Formal Validation (011)
Mathematical techniques for proving program correctness.

Formal testing (001)
Testing conducted according to test procedures which are
documented and approved by contractor and customer.

Formal testing (009)
Testing performed in accordance with customer-approved test
plans. This type of testing verifies that the software system

is operating according to the requirements of the development specifications. Formal testing is usually performed during the system evaluation phase of software development. Terms used in the literature to describe this testing include:
1 System Integration Testing
2 Prototype Testing
3 System Testing
4 Acceptance Testing

Function (001)
A grouping of routines which performs a prescribed function.

Function (012)
A sub-division of processes.

Functional requirements (006)
Describe the functions the system must perform.

Functional specifications (012)
Describe a system in terms of its principle functions and their interrelationships, i.e., the functional relationships of the parts.

Functional testing (009)
The execution of independent tests designed to demonstrate a specific functional capability of a program or a software system.

Generator (003)
A generator produces test data or test cases to exercise the target system. A generator in this case is differentiated from a simulator because it actually creates test data using numerical integrators, random number generators, etc. Once the data are produced by the generator, a simulator might be required to route the data to the system.

Generators are useful in a system test environment where "live" data is not available. Useful output of a data generator are tapes of logged data that can be used with a data replay facility for establishing standard test cases.

HIPO (008)
Hierarchy plus input/process/output is a graphic design technique used to show function. HIPO diagrams describe functions in terms of the input to a process. They show a system, subsystem, or program functionally, i.e., the functions that it performs, answering the question "What does it do?" Since these diagrams are visual, they are easier to understand than most documentation which is narrative. Although flowcharts are another graphic design technique, they show organization and logic in contrast to function.

Hardware monitors (003)
A unit that obtains signals from a host computer system through probes attached directly to the computer's circuitry. The signals obtained are fed to counters and timers and are recorded. These data are reduced to obtain information about CPU utilization, channel activity, etc. These data can be used to improve both system and program performance.

Implementation misinterpretation error (018)
An error for a unit of source code associated with a program due to the mininteroretation of the program specifications.

Implied system performance (002)
An unwritten requirement which is understood by the majority of the project team to be essentially equivalent to a written requirement.

Informal proof of correctness (012)
The visual inspection of a small, comprehensive set of test cases indicating that the code of a program segment matches its specification. Validation of the program segment is based on axioms stating that lower-level segments match their specifications.

Informal testing (009)
Testing that utilizes internal test documentation control and

procedures. Informal testing usually is designed to be development group testing and requires no formal customer approval. Informal testing usually begins when the first program unit is coded and continues throughout the system implementation phase of software development. Terms used in the literature to describe this testing include:

1 Unit Testing
2 Subsystem Testing
3 Integration Testing
4 Component Testing
5 Development testing

Information (018)
Correlation of data for the process of informing.

Instruction simulator (003)
A computer program used to simulate the execution characteristics of a target computer using a sequence of instructions of a host computer. The instruction simulator provides bit-for-bit fidelity with the results that would be produced by the target computer following the same operations and initial conditions.

Instruction trace (003)
A computer program used to record every time a certain class of operations occurs and trigger event-driven data collection. In some cases, this creates a complete timed record of literally everything significant that occurred during program execution. These traces contain data on instruction and become a permanent record of a program's execution.

Instructions (001)
Machine instructions (machine dependent parameter)

Instrumentation tools (012)
Those programs that monitor and record information about an object system, or portions thereof, as it operates data reduction and analysis.

Interface checker (003)
A computer program used to automatically check the range and limits of variables as well as the scaling of source programs to assure format compliance with interface and control documents.

Interface specification document (012)
A document that serves as a communications vehicle between the Configuration Control and Technical Implementation processes, and supports the coordination of efficient, controllable interfaces.

Internal delivery (001)
The point at which the sofware as an entire package is given to the independent test group.

Interrupt analyzer (003)
A computer program that examines source code and determines potential conflicts in the use of data and/or storage due to interrupts.

JOVIAL (012)
A programming language designed specifically for command and control projects.

Job (018)
Computer job consisting of one or more steps such as compilation, assembly, or utility runs.

Language processors (003)
Computer programs used to translate high-level or symbolic instruction mnemonics into computer-oriented code capable of being obeyed by a computer. These processors typically have capabilities for error detection through syntax analysis and provide symbolic addressing, expression evaluation, and symbol cross-reference listings. Compilers, assemblers and meta-assemblers are examples of this category of aids.

Levels of abstraction (012)
A design and implementation method that helps produce reliable software that is more easily modified and maintained

by identifying and placing into a hierarchical structure the functional processes and data resources that constitute the system's program structure. In addition, a design discipline that supports the creation of a well behaved, hierarchically structured, modular syste.

Line of source code (013)
80 character card image of source code.

Lines of source code from another source (018)
Code not developed but extracted from other sources.

Loadable program data (012)
Data that is relocatable or absolute binary modules produced by a link editor.

Loader (003)
A computer program that enables external references of symbols among different assemblies as well as the assignment of absolute addresses to relocatable strings of code. This program provides diagnostics on assembly overlap, unsatisfied external references, and multiple defined external symbols.

Logic equation generator (003)
A computer program used to automatically reconstruct arithmetic text and to flowchart assembly language programs. One such program translates assembly language instructions into a machine-independent microprogramming language and builds the microprogramming statements into a network in which flow of control is analyzed and equations reconstructed.

MAP program (003)
A computer program used to provide location and/or size information about all or selected parts of the target system, or about device-resident data.

Major error (002)
A catastrophic event which interrupts or could interrupt most of all major system functions, e.g. an infinite loop, system crash, a major memory overflow, a data base corruption, etc.

Management (012)
A term that indicates methodology, tools, and procedures.

Management control and project visibility (012)
Those processes that monitor the project's status in respect to planned levels of schedule, cost, and performance, and take corrective action if necessary.

Management functions (006)
Although the management process has been described in many ways, four basic functions have received general acceptance - planning, organizing, controlling and communicating.
1. Planning. The function to determining the project objectives and the policies, programs, procedures and methods for achieving them. The planning function must provide a framework for decision making.
2. Organizing. The function of determing the activities required to achieve the objectives of a programming project, the departmentation of these activities and the assignment of authority and responsibility for their performance.
3. Control. The function of assuring that the various components of a project are performing in accordance with the plan. Control is essentially the measurement and modification (if necessary) of component activities to assure the accomplishment of the overall plan.
4. Communications. The function of transferring information among decision makers throughout the project.

Management statistical data (018)
General name applied to all the data collected and accumulated by the PSL for the purpose of producing management reports including both Plan and Actual data.

Management statistical data base (018)
A data base containing Management Statistical data for an ongoing programming project.

Manual based testing (009)

Testing that is usually directed at evaluating both the design and the product (i.e., programs and documentation). The design is usually evaluated from documents containing information such as functional requirements, system specifications, and program specifications. The product evaluation usually involves review of the computer programs and the documentation describing the programs or systems.

Metacompiler (012)
A compiler system designed specifically to implement (compile) language compilers.

Metric (001)
A measure of the extent or degree to which the software possesses and exhibits a certain characteristic, quality, property, or attribute.

Minor error (002)
A marginal event which allows or could allow some portions of the system to operate properly while interrupting others, e.g. some missing output, some wrong output, an inaccurate computation, a recoverable transient error, etc.

Mistake (011)
A human action producing an unintended result.

Modeling and simulation tools (012)
Tools used for trade-off studies and to investigate particular abstractions and approaches for the system design. They are useful for analyzing and modeling particular approaches to system designs. Examples include:CASE, CSS, GPSS, MODLIT, SCERT, and SPCL.

Modifiability (013)
Implies controlled change, in which some parts or aspects remain the same while others are altered, all in such a

way that a desired new result is obtained.

Modular programming (003)
the technique of producing relatively small, easily interchangeable computer routines which meet certain standardized interface requirements. This technique makes it easier to develop and verify completed computer programs. Modularity is accomplished by breaking the program into limited line-segments that perform complete functions and are therefore completely understandable in themselves. Aids that help implement these techniques are standards and procedures.

Modularity (013)
Deals with how the structure of an object can make the attainment of some purpose easier. Modularity is purposeful structuring.

Module (012)
A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading.

Object program date (012)
The resulting form of a source language program after processing by a compiler or assembler. They are also called Object Modules.

The object program is in a format suitable for loading and execution. It may require additional processing by a link loader or link editor.

Operational (001)
The status given a software package once it has completed contractor testing and its turned over to the eventual user for use in the applications environment.

Overlay program (003)
A computer program that allows specific system components (load modules, core, data base, etc.) to be modified during execution in the case of modules, a program with an error can be replaced in core without bringing the system down and starting it up again. System parameters that affect performance can be varied during execution to compare vearious priority, timing, etc., schemes.

PSL job (018)
All of the computer processing that results from a single user request to execute the PSL for the purpose of performing one or more PSL functions such as update, compile or output.
Path analysis (007)
A software technique which scans source code in order to design an optional set of test cases to exercise the primary paths in a software module.
Performance (009)
The evaluation of nonlogical properties (i.e., computer run time,

resource utilization) of a software system. Performance is measured in terms of the amount of resources required by a software system to produce

a result.
Performance oriented data (012)
Information regarding management items.
Performance requirements (006)
Specify the time and space constraints which must be met.
Peripheral simulator (003)
A computer program used to test critical computer/peripheral interfaces that exist in real-time applications. These simulators range from functional, in which case the peripheral provides feedback to the program on the assumption that all interface constraints are satisfied, to high-fidelity simulations in which the interface constraints must be modeled to a detailed level of timing and message formatting.
Plan data (018)
Data describing the method or scheme of action for a project that will be included in the management reports.
Planning (012)
A technique that includes the evaluation of project requirements in terms such that logical assignments can be made. Also see Planning and Scheduling.
Planning and scheduling (012)
All activities that include an evaluation of the project's requirements and making assignments to conduct the project.
Pre-execution tools (012)
Tools that operate on the linguistic description of a program and do not require its execution. Examples include: syntax checking, interactive compilers, program reference listings, flow charts, and reformatters.
Precompiler (007)

A particular type of computer program which has the following characteristics:
1. It is normally executed immediately preceding a program compilation.
2. Its input consists of programming statements, of which all or part are unacceptable to the compiler.

3. It generates, as output, a computer program in a syntax acceptable to the compiler.
Process construction (003)
A technique used to combine and link independently-coded modules into a run-time process. These include linkages to the operating system. The technique allows for rapid reconfiguration based on stimuli from the run-time environment of a software system to reflect changes made to a number of its modules. Specific computer programs are available that serve as aids to implementation These include special-purpose editors and control programs.
Product (012)
Everything contracted for, produced for, and delivered to the customer. Examples include: hardware, programs, documents, and training.

Product certification (012)
A demonstration that the actual system performance corresponds to the expected system performance.
Product warranty (012)
A process that precedes configuration control and quality assurance activities. This process is a device for customer feedback after a software system has been delivered. It provides: (1) a means by which the customer reports suspected problems; (2) a means for technical and contractual evaluation of these problems; (3) arbitration and appeal procedures; and (4) a means to re-certify the corrected system. Also see Product Warranty and Maintenance, and Product Warranty Procedures.
Product warranty and maintenance (012)
A device, indicating the procedures to be followed during the warranty period of a system, for customer feedback after the delivery of a system.
Product warranty procedures (012)
A customer feedback device that precedes the delivery of a product,

and indicates the procedures to be followed during the warranty period of a system.
Production libraries (003)
A technique used to provide constantly up-to-date representations of the computer programs and test data in both computer and human readable forms. The current status and past history of all code generated is also maintained. Specific library programs are available to serve as aids to implementation.
Production run (012)
The operation of a software system under real operating conditions and the production of useful products for the customer. This is contrasted with a test run, which is the operation of a software system to test its performance.
Production tools (012)
Tools related to the specific requirements for software system production. Examples include: Structured Programming Design Tool, Program Production Library, Program Validation Tools, Program Stub Simulators, compilers, link editors, and source program editors. See Program Production Tools.
Productivity (006)
Traditionally, the generally accepted description (if not definition) of programming productivity has been "lines-of-code/man-month" (i.e., quantity of code produced . Contemporary research suggests that the definition of productivity should somehow contain at least three additional elements:

1. A qualitative element concerned with the correctness and efficiency of a program.
2. A qualitative element concerned with complexity (i.e., the difficulty of comprehending the application (or algorithm) being implemented and the organization of the program hierarchial tree structure (e.g., number of levels, width, data interfaces).
3. An element concerned with the cost of the program.

Program (018)
The lowest level of module that can be assembled or compiled and can be executed as a single entity.
Program Office (003)
The field office organized by the Program Manager to assist him in accomplishing the program tasks (AFR 800-2).
Program description specifications (012)

These are informational documents produced for the customer, usually according to his requirements.
Program design language (005)
(PDL) a design tool used to facilitate the translation of functional specifications into computer instructions.
Program development tools (012)

Tools that take the computer-stored source program into an object
code module, then a load module, and subsequently execute the
code in a specific test environment.
Program flow analyzer (003)
A computer program that provides statistics on source code
statement usage and timing data on program elements during
test case executions.
Program implementation (012)
All activities associated with software module design, coding
and testing and the integration of software modules into a
functional system operating on the total, final hardware
configuration.
Program instrumentation (012)
A quantitative assessment of how thoroughly a program is
exercised by a set of test cases.
Program instrumentation tools (012)
Tools that provide a mechanism for monitoring and recording
information about an object program as it operates. Examples
include traces and snapshot dumps.
Program management directive (003)
The official HQ USAF management directive used to provide
direction to the implementing and participating commands and
satisfy documentation requirements. It will be used during the
entire acquisition cycle to state requirements and request
studies as well as initiate, approve, change, transition,
modify, or terminate programs. The content of the PMD, including
the required HQ USAF review and approval actions, is tailored
to the needs of each individual program (AFR 800-2).
Program management plan (003)
The document developed and issued by the Program Manager that shows the
integrated time-phased tasks and resources required to complete the
task specified in the PMD. The PMP is tailored to the needs of
each individual program (AFR 800-2).
Program manager (003)
The generic term used to denote a single Air Force Manager
(System Program Director, Program/Project Manager, or System/Item
Manager) during any specific phase of the acquisition life cycle
(AFR 800-2).
Program production tool (012)
An established procedure or computer program that provides
assistance in the development, implementation, and testing
of a software system.
Program references (012)
Special listings, produced by many compilers, that increase the
user's understanding of the nature of the program produced.
Examples include dictionary listings, core-storage maps, and
cross-reference listings.
Program segment (012)
A combination of program steps and calls to lower-level
program segments.
Program sequencer (003)
A computer program used to force execution of all possible
program instructions and branches to determine program flow, execute
seldom-used branches, and to verify proper program operations.
The aid is often used with an instruction simulator.
Program stub (018)
A temporary (i.e., dummy) unit of source code which
is part of an incomplete structured program and will be
replaced by the actual unit of code when it is completed.
Program stub simulators (012)
Generalized subroutines used in Program Stubs that supply
the code required to establish the desired linkage with the
higher-level program segments. They include generalized table
lookup routines, random number generators, or routines that only
record the invocation of a program segment.
Program stubs (012)

Dummy program segments containing enough code to establish linkage with a higher-level segment.

Program validation (012)
All techniques used to ensure correct programs, including system and sub-system tests and system integration testing.

Program validation tools (012)
Tools that are used in the preexecution and run time phases of Program Implementation and Program Validation.

Programming support library (005)
(PSL) a repository for data necessary for the orderly development of computer programs using structured programming technology. The data repository is in two forms: data is stored in machine readable form accessible by the computer and the identical data is stored in hard copy form in project notebooks. A PSL also includes the necessary computer and office procedures for manipulating this data.

Project (012)
All processes necessary to produce a product.

Project (019)
A general term used to describe a software development effort.

Project construct data (012)
Information on design and implementation details.

Project data base (012)
A project-specific catalog containing management data and program data supplied and used by various facilities. Examples include: the contents, names, and linkages of all the modules in a particular system. The management data pertaining to the tasks and milestones of a specific project.

Proof of correctness (012)
Proof that a program produces correct results for all possible inputs. Validation of a program in the same way a mathematical theorem is proved correct, i.e., by mathematical analysis of its properties.

Proofs of correctness (017)
An alternative to executing tests of software to demonstrate its correctness is the method of analytic proofs. The verification process consists of making assertions describing the state of a program initially, at intermediate points in the program flow, and at termination, and then proving that each assertion is implied by the initial or prior one and the transformations performed by the

program between each two consecutive assertions. An assertion consists of a definition of the relationships among the variables at that point in the program where the assertion is made. The proofs employ standard techniques for proving theorems in the first order predicate calculus. Proof of the correctness of a program using this approach obviates the need for executing test cases, since all possibilities are covered by the proofs.

Quality assurance (012)
The process of activity during which the system design is audited to determine whether or not it represents a verifiable and certifiable specification, and during which test plans and test procedures are formulated and implemented. This activity ensures the technical compliance of the software system—a product—to its Requirements and Design specifications. Quality Assurance is an independent audit review of all products to

ensure their compliance to a management-directed standard of quality.

Record generator (003)
A computer program used to construct test data. Essentially, the program contains a library of data formats, including the location, size, character (alpha or numeric), and normal contents of each field of each record type from which it generates

records required for testing.
Reporting (012)
A major sub-division within Configuration Control. The reporting and documenting activites needed to monitor the status of configuration during the life of a system.
Requirements analysis (012)
The process of studying the customer's problem from both that of the developer and the user in order to arrive at a

functional definition of system requirements. Includes all activities related to analyzing and developing a clear, unequivocal, and mutually agreed upon set of functional

specifications for a project.
Requirements and design specifications (012)
Precise specifcations of the operational characteristics of the final product.
Requirements language (003)
A computer program used to provide a succinct and unambiguous specification of the system, then computer requirements. It more precisely allows requirements to be communicated and translated in a hierarchical manner.
Routine (001)
Smallest group of compilable code.
SNAP generator (003)
A computer program used to search a tape to provide absolute program locations in memory that are relative to program labels. In addition, the computer program provides storage address and binary scaling associated with user-specified variable names. A SNAP generator is used typically to present a picture of a selected portion of memory.
Schedule management (012)
A method of determining what work must be done to produce each element of a system.
Scoring program (003)
A computer program that performs the same calculations as the target system. The program includes a compare capability so that results computed by the target system can be automatically

compared and discrepancies flagged.
Simulator (003)
A computer program that provides the target system with inputs or responses that resemble those that would have been provided by the process for the device being simulated. The simulator's function is to present data to the system at the correct time and in an acceptable format. this is a category for generalized simulators that cannot be classified according to specific functions under designators 15, 17, 18, 24, 34, 49, 50 or 51.
Snaps (003)
A computer program that allows intermediate data values to be recorded on an external medium during execution. Snaps are usually formatted prior to being recorded so that no postprocessing is required. Snaps are assembled in-line with the application code and therefore cannot be dynamically overridden during execution except for being turned on or off.
Software (016)
Computer program code and its associated necessary data and documentation.
Software development cycle (006)
The software development cycle consist of four phases: definition, design, implementation and evaluation.
Software development cycle (014)
1. ; requirements specifcation. Translation of an operational (or application) requirement into a statement of the functions to be performed.

2. System design. Translation of the requirements into a description of all the components necessary to implement the

system.
3. Programming. Production of the code which will control the system and perform all required logic and computation.
4. Checkout. Verification that the coded and other components of the system satisfy the original requirements.

Software development process (016)
The formal process by which project objectives are transformed into project requirements, then into design specifications, implemented into code, tested, and finally placed and maintained in operational status.

Software errors (011)
Any discrepancy between a computed, observed or measured quantity and its true, specified, or theoretically correct value. Errors are introduced into software by human mistakes that is: deficiencies or

misinterpretations of design criteria, logical mistakes, syntatical mistakes made in transcribing program statements into the input data.

Software factory (012)
A collected set of tools, methodologies, and a common data base that provide a procedural approach to the successful completion of software projects.

Software failure (011)

Software monitor (003)
A computer program that provides detailed statistics about system performance. Because software monitors reside in memory, they have access to all the tables the system maintains. Therefore, they can easily examine such things as core usage, queue lengths, individual program operation, and so on to help measure performance.

Software product (012)
The software component of the product.

Software reliability (002)
Software reliability is defined as the probability that a given software program operates for sometime period, without an external software error, on the machine for which it was designed given that it is used within design limits.

Software reliability measures (011)
Probability measures of the quality with which design requirements have been transformed into software programs. A typical measure is the mean time to failure as a function of operating time.

Software system (012)
See Software Product.

Software system dependability (011)
The probability that the application program together with its supervisory program, data base and hardware will perform in its intended environment. The environment will include anomalies and failures, such as:
1. Deficiencies in requirements
2. Software design errors (incorrect algorithms, word length problems, timing problems, etc)
3. Software failures
4. Processor errors
5. Memory errors
6. Failures in the communication network
7. Failures in peripherial devices
8. Operator mistakes
9. Power failures
10. Environmental failures
11. Gradual erosion of the data base
12. Hardware saturation (CPU, memory, I/O channels)

Software testing (011)
The process of exercising software in an attempt to detect errors which exist in the code. Software testing does not prove that a program

is correct.

Source unit end date (018)

The date of the last update made to the unit of source code.

Source unit start date (018)

The date the unit of source code was entered into the PSL.

Specification (012)

The technical definition of the system and its parts.

Specification omission error (018)

An error for a unit of source code associated with a program due to omissions in the program specifications.

Specified performance requirements (002)

A written requirement, figure of merit, or parameter which qualitatively or quantatitively defines system performance.

Standards (003)

Procedures, rules, and conventions used for prescribing disciplined program design (program structuring, and data structuring) and implementation. Architecture and partitioning rules, documentation conventions, configuration and data management procedures, etc., are among those standards to be disseminated.

Standards enforcer (003)

A computer program used to automatically determine whether prescribed programming standards and practices have been adhered to. The program can check for violations to standards set for such conventions as program size, commentary, structure, etc.

Statements (001)

Programming language at the source code level.

Statistical prediction (009)

The computation of a confidence factor that indicates the effectiveness of the programming and verification process by inserting errors into the software system.

Stepwise refinement (004)

The process of defining data in more and more detail as the need arises during the programming process.

Structured program (005)

A program constructed of a basic set of control logic figures which provide at least the following: sequence of two operations, conditional branch to one of two operations and return, and repetition of an operation. A structured program has only one entry and one exist point. In addition, a path will exist from the entry to each node and from each node to the exit.

Structured program segments (012)

A combination of program steps and calls to lower-level program segments.

Structured programming (003)

The technique used in structured programming (limited number of logic structures, top-down development, etc.) make it easier to develop and verify completed computer programs. Aids that help implement these techniques follow.

1. Automated language restructurer - a computer program used to restructure source programs into an acceptable structured form.

2. Language enhancer - a computer program used to provide existing high-order language compilers with acceptable language constructs for structured programming. Language enhancers include both the preprocessor and macro mechanizations presently in use.

3. Source code indenter - a computer program used to automatically indent source code listings to add to their readability.

4. Production support libraries - a form of production library used to record and store programming data.

Structured programming (005)

(SP) the process of developing structured programs. Associated with structured programming are certain practices such as indetationsof source code to represent logic levels, the use of intelligent data names and descriptive commentary.

Structured programming (012)
A programming discipline providing a means of expressing a system design that ensures a testable and understandable implementation and that enforces simple and well-defined corrections between

program modules. The programming discipline uses the repeated application of a small number of basic control statements to form simple program constructs that represent large and complex programs. The process of creating the program modules includes: making local or tactical programming decisions within the designed module: writing program segments that represent these decisions: and, integrating program segments into a unit corresponding to a system module.

Structured programming (015)
Structured programming is strictly more of a program organization discipline than a design technique, but it can be used to significantly enhance most of the available design techniques. It is basically a set of standards for organizing the control structure of a set of computer programs. The key ideas in structured programming are:
1. Only "proper programs" having one flow of control in and out of each unit should be developed.
2. Only three basic control structures are allowed: DOWHILE, IFTHENELSE, and SEQUENCE. These three structures are sufficient to express any proper program(15). Two additional structures are optional, the DOUNTIL and the CASE.
3. Programs are organized according to a hierarchical, modular block structure.
4. Additional standards are imposed on the size of modules, the formatting of instructions and declarations, program commentary, etc.
Boehm, Barry W.

Structured programming language (012)
A language that supports the concepts of Structured Programming and permits closer integration of management and production tools.

Structured programming technology (005)
A term which collectively references the following list:

1. Program design language
2. Programming support library
3. Top down structured programming

Structured programs (012)
Small computer programs written in a restricted, simple syntax.

Structured segment (005)
A logically complete set of executable instructions constructed of nested structured programming figures. In addition to or in place of executable instructions, a structured segment may include non-executable instrumentions such as data declarations and descriptive commentary.

Structured source code listing (005)
A listing for a top down structured program (TDSP) comprised of the following sections:
1. Section 1 contains the first executable structured segment (commonly referred to as the top level segment as coded in the source programming language.
2. Section 2 contains all remaining structured segments. the structured segments are alphabetized by name. As in Section 1 each structured segment is represented as coded in the source programming language.
3. Section 3 contains the executing sequence among the structured segments.

Structured walk-through (009)
A generic name given to a set of techniques (i.e., design validation, design verification, and code verification), each with different objectives and each occurring at different times in the software development cycle.
Subsystem (018)
A subdivision of a software system with a meaningful product to the user and is comprised of one or more programs. A subsystem name identifies the top unit of a tree structure.
System (018)
Consists of more than one software subsystem and provides a solution to a problem.
System design (012)
The process of transferring the design requirements into an overall system structure that supports programs satisfying those requirements. Includes all activities concerned with transforming functional requirements/specifications into a Structured Programming system capable of satisfying those requirements.
System simulations (003)
Computer system simulation is a technique used to predict system performance by exercising a model of the system hardware/software over time. Simulation based on well-planned experiments representative of the real-world environment will produce results that help verify and improve system performance. The simulations are also used to help predict how the system will react to alternative loads with modified configurations. Specific language systems such as ECSS, CSS, SCERT, and SAM have been devised to act as aids to implementation.
System verification (012)
All activities concerned with ascertaining that the system performs as the customer intended.
TRACE program (003)
A computer program that records the chronological sequence of events taken by a target program during its execution.
Task milestone (012)
A major visible event or interface during a project.
Terminal simulator (003)
A computer program used to present input messages to the control program so as to appear that they had been input from an actual terminal device.
Test (003)
Any program or procedure that is designed to obtain, verify, or provide data for the evaluation: research and development

(other than laboratory experiments); progress in accomplishing development objectives; or performance and operational capability of systems, subsystems, components, and equipments items (AFR 80-14).
Test beds (003)
A test site that either contains the actual hardware and interfaces (hardware test bed) or simulates them (software test bed).
1. Hardware test bed - includes actual computer and interface hardware, thus permitting actual checkout of hardware/software interfaces and actual input/output. The program execution is confirmed using actual hardware timing characteristics, but the output is limited, and it has limited diagnostic capabilities.
2. Software test bed - uses an instruction simulator to simulate actual hardware. The approach poorly represents actual I/O, runs 7 to 15 times real-time, and is an expensive method of

conducting lengthy testing of software. The approach permits full control of inputs and computer characteristics, allows processing of intermediate outputs without destroying real time, and allows full test repeatability and diagnostics.

Test data (012)
    Test environment data that are prepared manually or by an automatic test case generator.

Test data base (009)
    Collection of data stored on a computer peripheral device (e.g., tape, disk that closely matches the "real" data base). Ideally, a test data base should be identical to a real data base but usually it only provides representative data.

Test drivers, scripts, data generators (003)
    To run tests in a controlled manner, it is often necessary to work within the framework of a "scenario"--a description of a dynamic situation. To accomplish this, it is necessary to load the input data files for the system with data values representing the test situation or events to yield recorded data to evaluate against expected results. These aids permit relatively easy generation of data in external form to be entered automatically into the system at the proper time.

Test plan (003)
    A management document that describes how and when specified test objectives will be met (AFR 80-14).

Test plan document (012)
    A plan that tests the effectiveness of the object system, as

    implemented, and determines how it can be validated, verified, and certified. This document is input to the subsequent functions of the Quality Assurance process and to the Program Validation function.

Test result processor (003)

    A computer program used to perform test output data reduction, formatting and printing. Some perform statistical analysis where the original data may be the output of a monitor.

Testing (009)
    Measures how well the specifications are met.

Text data (012)
    Program documentation that is prepared manually and updated by a text editor.

Timing analyzer (003)
    A computer program that monitors and prints execution time of all program elements (functions, routines, and subroutines).

Top down design (017)
    Top down design implies an ordering to the sequence of decisions which are made in the decomposition of a software system, by beginning with a simple description of the entire process (top level). Through a succession of refinements of what has been defined at each level, lower levels are specified.

Top down programming (005)
    The concept of performing in hierarchical sequence a detailed design, code, integration and test as concurrent operations.

Top down programming process (017)
    An expansion of functional specifications to simpler and simpler functions until, finally, statements of the programming language itself are reached.
    Ref H. Mills, "Top Down Programming in Large Systems", "Debugging Techniques in Large Systems", R. Austin, (ed), Prentice-Hall, NJ 1972, 41-55.

Top down structured program (005)
    (TDSP) a structured program with the additional characteristics of the source code being logically, but not necessarily physically, segmented in a hierarchical manner and only dependent on code already written. Control of execution between

segments is restricted to transfers between adjacent hierarchical segments.

Top down structured programming (005)
The process of developing top down structured programs. Associated with top down structured programming are certain practices such as indentations of source code to represent logic levels, the use of intelligent data names and descriptive commentary. Top down structured programming requires top down programming as the primary implementations methodology.

Top down testing (017)
If modules are produced in a top down order, then top down testing can also be employed using a partially completed system in which lower levels are represented by program "stubs", and programs are executed in the environment in which they will actually operate. This approach allows for earlier integration and testing which should uncover problems sooner then conventional testing where integration is the last step.

Top-down program development (012)
The process of coding and testing Structured Programs successively downward from the top level of program design to the bottom levels, continuously exercising the actual interfaces between program

modules. The bottom level may be standard packaged routines--data access methods, sort routines or interfaces with other systems. This approach is the opposite of the usual one of checking the bottom-level modules first and working up, finally integrating and testing the entire system. Here, the integration of modules is a continuous process.

Tracer program (007)
A program analysis tool which will analyze computer programs looking for

code" - i.e., code in a program which can not be executed.

Translator (003)
A computer program written to accept information from one system of representation and convert this into equivalent information in another system of representation.

Traps (003)
A technique in which the logic flow of a program is interrupted for the purpose of setting aside interim results for test measurement or performing special test functions.

Unit (018)
a named subdivision of a program which is capable of being stored in a PSL and manipulated as a single entity.

Units consistency analysis program (003)
A computer program that analyzes source code to verify units consistency for each usage of each parameter.

User (001)
The individual at the man/machine interface who is applying the software to the solution of a problem, e.g. test or operations.

Utilities (003)
Computer programs employed by other V/V/C aids to provide special services. These services include preparing program deck listings, creating load tapes, and plotting output results.

Utilities (012)
Tools that facilitate the production and control of software systems. These include tools for data set management, program development, and program execution.

Validation (009)
The process of determining whether executing the system (i.e.,

software, hardware, user procedures, personnel) in a user environment causes any operational difficulties. Validation is more difficult than the verification process since it involves questions of the completeness of the specification and environment information. There are both manual and computer based validation techniques.

Validation (012)

The process of ensuring that specific program functions meet their detailed design requirement specifications. Also, see program Valiuation, Program Validation Tools, and Validation and Debugging

Tools.

Validation and debugging tools (012)
Tools related to the production of correct, serviceable programs. Validation includes the prevention, detection, diagnosis, recovery, and correction of errors. Debugging includes correcting errors of both a logical and a clerical nature.
Validation criteria (012)
A guide defining what will be used to determine completion of a milestone. This includes successful operation of a test tool,

or acceptance of a document, or approval by the review board.
Validation tools (012)
See Validation and Debugging tools.
Verification (009)
The process of determining whether the results of executing the software product in a test environment agree with the specifications. Verification is usually only concerned with the software's logical correctness (i.e. satisfying the functional requirements) and may be a manual or a computer based process (i.e., testing software by executing it on a computer).
Verification (012)
The process of ensuring that the system and its structure meet the functional requirements of the Baseline Specification

Document. Also see System Verification.
Verification validation certification (003)
(of computer programs). The process of determining that the computer program was developed in accordance with the stated specification and satisfactorily performs, in the mission environment, the function(s) for which it was designed (AFR 800-14). See computer program verification, computer program validation, computer program certification.
Version modification level (018)
An indication of the version and modification level of a unit of source code.

METRIC SYSTEM

BASE UNITS:

Quantity	Unit	SI Symbol	Formula
length	metre	m	...
mass	kilogram	kg	...
time	second	s	...
electric current	ampere	A	...
thermodynamic temperature	kelvin	K	...
amount of substance	mole	mol	...
luminous intensity	candela	cd	...

SUPPLEMENTARY UNITS:

plane angle	radian	rad	...
solid angle	steradian	sr	...

DERIVED UNITS:

Acceleration	metre per second squared	...	m/s
activity (of a radioactive source)	disintegration per second	...	(disintegration)/s
angular acceleration	radian per second squared	...	rad/s
angular velocity	radian per second	...	rad/s
area	square metre	...	m
density	kilogram per cubic metre	...	kg/m
electric capacitance	farad	F	A·s/V
electrical conductance	siemens	S	A/V
electric field strength	volt per metre	...	V/m
electric inductance	henry	H	V·s/A
electric potential difference	volt	V	W/A
electric resistance	ohm		V/A
electromotive force	volt	V	W/A
energy	joule	J	N·m
entropy	joule per kelvin	...	J/K
force	newton	N	kg·m/s
frequency	hertz	Hz	(cycle)/s
illuminance	lux	lx	lm/m
luminance	candela per square metre	...	cd/m
luminous flux	lumen	lm	cd·sr
magnetic field strength	ampere per metre	...	A/m
magnetic flux	weber	Wb	V·s
magnetic flux density	tesla	T	Wb/m
magnetomotive force	ampere	A	...
power	watt	W	J/s
pressure	pascal	Pa	N/m
quantity of electricity	coulomb	C	A·s
quantity of heat	joule	J	N·m
radiant intensity	watt per steradian	...	W/sr
specific heat	joule per kilogram-kelvin	...	J/kg·K
stress	pascal	Pa	N/m
thermal conductivity	watt per metre-kelvin	...	W/m·K
velocity	metre per second	...	m/s
viscosity, dynamic	pascal-second	...	Pa·s
viscosity, kinematic	square metre per second	...	m/s
voltage	volt	V	W/A
volume	cubic metre	...	m
wavenumber	reciprocal metre	...	(wave)/m
work	joule	J	N·m

SI PREFIXES

Multiplication Factors	Prefix	SI Symbol
1 000 000 000 000 = 10^{12}	tera	T
1 000 000 000 = 10^{9}	giga	G
1 000 000 = 10^{6}	mega	M
1 000 = 10^{3}	kilo	k
100 = 10^{2}	hecto*	h
10 = 10^{1}	deka*	da
0.1 = 10^{-1}	deci*	d
0.01 = 10^{-2}	centi*	c
0.001 = 10^{-3}	milli	m
0.000 001 = 10^{-6}	micro	µ
0.000 000 001 = 10^{-9}	nano	n
0.000 000 000 001 = 10^{-12}	pico	p
0.000 000 000 000 001 = 10^{-15}	femto	f
0.000 000 000 000 000 001 = 10^{-18}	atto	a

* To be avoided where possible

BEST AVAILABLE COPY

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

AMERICAN REVOLUTION BICENTENNIAL 1776-1976